

# Using Large Language Models for Knowledge Engineering (LLMKE): A Case Study on Wikidata

Bohui Zhang<sup>1</sup>, Ioannis Reklós<sup>1</sup>, Nitisha Jain<sup>1</sup>, Albert Meroño Peñuela<sup>1</sup> and Elena Simperl<sup>1</sup>

<sup>1</sup>Department of Informatics, King's College London, London, UK

## Abstract

In this work, we explore the use of Large Language Models (LLMs) for knowledge engineering tasks in the context of the ISWC 2023 LM-KBC Challenge. For this task, given subject and relation pairs sourced from Wikidata, we utilize pre-trained LLMs to produce the relevant objects in string format and link them to their respective Wikidata QIDs. We developed a pipeline using LLMs for Knowledge Engineering (LLMKE), combining knowledge probing and Wikidata entity mapping. The method achieved a macro-averaged F1-score of 0.701 across the properties, with the scores varying from 1.00 to 0.328. These results demonstrate that the knowledge of LLMs varies significantly depending on the domain and that further experimentation is required to determine the circumstances under which LLMs can be used for automatic Knowledge Base (e.g., Wikidata) completion and correction. The investigation of the results also suggests the promising contribution of LLMs in collaborative knowledge engineering. LLMKE won Track 2 of the challenge. The implementation is available at: <https://github.com/bohuizhang/LLMKE>.

## 1. Introduction

Language models have been shown to be successful for a number of Natural Language Processing (NLP) tasks, such as text classification, sentiment analysis, named entity recognition, and entailment. The performance of language models has seen a remarkable improvement since the advent of several LLMs such as ChatGPT<sup>1</sup> and GPT-4 [1] models from OpenAI, LLaMa-1 [2] and Llama 2 [3] from Meta, Claude<sup>2</sup> from Anthropic, and Bard<sup>3</sup> from Alphabet.

This surge in the development and release of LLMs, many of which have been trained with Reinforcement Learning with Human Feedback (RLHF), has allowed users to consider the LMs as *knowledge repositories*, where they can interact with the models in the form of ‘chat’ or natural language inputs. This form of interaction, combined with the unprecedented performance of these models across NLP tasks, has shifted the focus to the engineering of the input, or the ‘prompt’ to the model in order to elicit the correct answer. Subsequently, there has been a steady increase in research outputs focusing on prompt engineering in the recent past [4, 5, 6].


*KBC-LM'23: Knowledge Base Construction from Pre-trained Language Models workshop at ISWC 2023*

✉ bohui.zhang@kcl.ac.uk (B. Zhang); ioannis.reklos@kcl.ac.uk (I. Reklós); nitisha.jain@kcl.ac.uk (N. Jain); albert.merono@kcl.ac.uk (A. M. Peñuela); elena.simperl@kcl.ac.uk (E. Simperl)

🌐 <https://bohuizhang.github.io/> (B. Zhang); <https://nitishajain.github.io/> (N. Jain); <https://www.albertmeronyo.org/> (A. M. Peñuela); <http://elenasimperl.eu/> (E. Simperl)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

<sup>1</sup><https://chat.openai.com/>

<sup>2</sup><https://claude.ai/>

<sup>3</sup><https://bard.google.com/>

Knowledge graphs (KGs) are a technology for knowledge representation and reasoning, effectively transferring human intelligence into symbolic knowledge that machines can comprehend and process [7, 8, 9]. The process of creating these KGs, referred to as knowledge engineering, is not trivial, either automatically or collaboratively within human communities [10]. Wikidata [11], as the largest open KGs, contains rich knowledge of real-world entities. It has been developed in a collaborative manner, with contributions from a community of users and editors [12].

While the concept of using LMs to construct and complete KGs has been extensively explored in previous research [13, 14, 15], the recent surge in LLMs performance has rekindled discussions about the possibility of leveraging the strengths of both technologies and unifying them [16]. Despite the immense potential offered by LLMs as knowledge bases, there exist fundamental disparities that differentiate them from KGs. The most pivotal of these distinctions lies in the domain of reasoning. Not only do traditional KGs store facts, they also impose logical constraints on the entities and relations in terms of defining the types of the entities as well as prescribing the domain and range of the relations. The capability of LLMs for logical reasoning remains unclear and appears to face challenges [17, 18]. Moreover, the most widely adopted and successful LLMs have been trained on data obtained from publicly available sources, and due to the inherent limitations of the training method of these models, they tend to exhibit expert-level knowledge in popular domains or entities while often displaying a limited understanding of lesser-known ones.

In this paper, we describe our approach LLMKE to using LLMs for Knowledge Engineering tasks, especially targeting solving the ISWC 2023 LM-KBC Challenge [19], and report our findings regarding the prospect of using these models to improve the efficiency of knowledge engineering. The task set by this challenge is to predict the object entities (zero or more) given the subject entity and the relation that is sourced from Wikidata. For instance, given the subject *Robert Bosch LLC* with Wikidata QID Q28973218 and the property *CompanyHasParentOrganisation*, the task is to predict the list of object(s), [*Robert Bosch*] and their matched QID(s), [‘Q234021’]. We used two state-of-the-art LLMs, gpt-3.5-turbo<sup>4</sup> and GPT-4 for this task. By performing different experiments using in-context learning approaches, we have been able to achieve a macro-average F1 score of 0.701, with F1-scores ranging from 0.3282 in the *PersonHasEmployer* property to 1.0 in the *PersonHasNobelPrize* property.

## 2. Related Works

### 2.1. LLMs for Knowledge Probing

The ability of LLMs to perform knowledge-intensive tasks, especially knowledge probing, has been extensively investigated [20, 21, 22]. In particular, several previous works have attempted to use language models to construct or complete KGs. Among early works, the LAMA paper by Petroni et al. [23] investigated the task of knowledge graph completion by probing LMs to extract facts via cloze-style prompts. Along similar lines, KG-BERT leverages the BERT language model to perform the link prediction task for knowledge graph completion[24]. The

---

<sup>4</sup><https://platform.openai.com/docs/models/gpt-3-5>

extent of the usefulness of LLMs for the construction and completion of knowledge graphs has since been further analyzed [13]. Follow up work after LAMA improved the performance even further [5, 20]. Recently, Veseli et al. [15] have performed a systematic analysis on the potential of LMs for automated KG completion. They report that LMs can be useful for predicting facts with high precision for some relations in Wikidata, though this is not generalizable. Prompt engineering has caught the attention of many recent works that aim to elicit knowledge from the language models [14]. These works are the most similar to our approach in this paper.

## 2.2. Knowledge Probing Benchmarks

To fulfil the need for comprehensively investigating the ability of LLMs to perform knowledge-intensive tasks, there has been a growing trend of knowledge-oriented benchmarks and datasets. These benchmarks encompass diverse domains, address various scenarios, including question answering, reading comprehension, and fact completion, and represent knowledge in different formats, including queries, cloze-style, incomplete triples, etc [21, 25]. And knowledge graphs, especially the large-scale and general-purpose ones, have become vital sources for constructing these benchmarks. As the pioneering dataset in the language models era, LAMA was constructed from a variety of knowledge graph sources of factual and commonsense knowledge, including T-REx [26], ConceptNet [27], etc. There are several benchmarks that evolved from it to overcome its limitations and expand its abilities, such as KAMEL [28] which extended LAMA from single-token objects to multi-token ones. KILT [29] was constructed from millions of Wikipedia pages spanning a wide range of knowledge-intensive language tasks. WikiFact [21] as a part of the HELM benchmark is the most similar to this challenge, where they use Wikidata relations and triples to construct the benchmark. But the challenge used a different evaluation paradigm. KoLA [30] aimed at measuring the real-world performance of LLMs by expanding beyond language modeling, adding evolving data sources, and attempting to measure the ability of the models in all facets of knowledge processing, ranging from knowledge memorization to knowledge creation. The data sources it used are also highly overlapping with Wikidata and Wikipedia.

## 3. Methods

### 3.1. Problem Formulation

Most of the previous works on using LLMs for fact completion stop at the string level, which leaves gaps for constructing hands-on knowledge graphs and thus hinders downstream application. Our work pushed a step forward on this task, where the extracted knowledge is not only in string format but also linked to their respective Wikidata entities. Formally, given a query consisting of subject entity  $s$  and relation  $r$ , the task is to predict a set of objects  $\{o_i\}$  with unknown numbers ( $|\{o_i\}| \geq 0$ ) by prompting LLMs and mapping the objects to their related Wikidata entities  $\{w_{o_i}, \dots, w_{o_n}\}$ .

## 3.2. The LLMKE Pipeline

### 3.2.1. Knowledge Probing

The pipeline consists of two steps: *knowledge probing* and *Wikidata entity mapping*. For the knowledge probing step, we engineered prompt templates for probing knowledge from LLMs. We adopt OpenAI’s gpt-3.5-turbo and GPT-4 in this step. For each of the LLMs, we run experiments with three types of settings. The first is question prompting, where LLMs are provided with questions as queries. For example, “Which countries share borders with Brazil?”. The second is triple completion prompting, where prompts are formatted as incomplete triples, such as “River Thames, RiverBasinsCountry:”. There are several heuristics employed in these two settings. For example, there are only 5 different Nobel Prizes, so *PersonHasNobelPrize* has 6 candidate answers, including the empty answer. When the answer space is limited, providing all potential answers in the prompt templates is likely to reduce the difficulty of formatting and disambiguating the objects, thus helping LLMs perform well.

In the third setting, we provide retrieval-augmented context to help LLMs by enriching knowledge from corpus, including Wikipedia and domain-specific websites. Trying to leave space for invoking the ‘critical thinking’ of LMs and for further investigating the effect of adding context, the prompts used in this setting are separated into two steps. At first, we ask LLMs to predict the objects based on their own knowledge using the same settings as question prompting. In the second step, we provided the context knowledge, and LLMs were asked to make predictions again by considering the context and comparing it with the previous response. The prompt is like ‘Given the context: [retrieval-augmented context], compared and combined with the previous predictions, [question prompt]’. In this case, we let LLMs to decide whether they will insist on their own knowledge or change their answers based on the context. In this study, we used Wikipedia as the general-domain context source. The first paragraphs of the entity’s Wikipedia page (the introduction) and the JSON format of the Wikipedia Infobox are organized and provided to LLMs. For relations that could potentially have empty results, the prompt indicated the required return format (i.e., [“”]).

In all settings, we perform few-shot learning, where we provide three examples (i.e., prompt and answer pairs) from the training set. Since the required format of results is a list, providing examples with the exact format is expected to help LLMs return better-formatted results.

### 3.2.2. Wikidata Entity Mapping

The entity mapping step first finds Wikidata entities for each object string using the MediaWiki Action API<sup>5</sup>. One of the actions, *wbsearchentities*<sup>6</sup> which searches for entities using labels and aliases, returns all possible Wikidata entities as candidates. Then, in the disambiguation step, the actual Wikidata entities linked to the objects are selected. The **baseline** disambiguation method selects the first entity from the list of candidates returned by the *wbsearchentities* action, which is notably incorrect. To reduce the cost while improving the accuracy for disambiguation, we treated different relations with three **improved** methods: *case-based*, *keyword-based*, and *LM-based*.

---

<sup>5</sup><https://www.wikidata.org/w/api.php>

<sup>6</sup><https://www.wikidata.org/w/api.php?action=help&modules=wbsearchentities>

The *case-based* method is a hard-coding solution for efficiently solving ambiguities for relations with smaller answer spaces and limited corner cases. It is built on the baseline method by adding the function that maps specific objects to their respective Wikidata QIDs. For example, *CompoundHasParts* only has all the chemical elements as its answer space. Further, there is only one mistake in the baseline method, ‘mercury’. Thus, when predicting for *CompoundHasParts*, the case-based method always maps ‘mercury’ in the object lists to Q925 (the chemical element with symbol Hg) instead of Q308 (the planet). For other relations with a larger answer space but also entities with common characteristics, we used the *keyword-based* method, which extracts the description of the candidate entities from its Wikidata page and searches entities with their description using relevant keywords. This method is used when there are common words in the entity description. For example, object entities of the relation *CountryHasOfficialLanguage* always have the keyword ‘language’ in their descriptions.

The above two methods clearly suffer from limitations due to their poor coverage and inflexibility. The third method is language model-based (*LM-based*). We constructed a dictionary of all candidate QIDs with their labels as keys and descriptions as values, concatenated it with the query in this first step, and asked LMs to determine which one should be selected. This method is used when there is no semantic commonality between the answers and disambiguation is required to understand the difference between entities, e.g., properties with the whole range of human beings as potential answers such as ‘*PersonHasSpouse*’. As there is no commonality among the labels and descriptions of answers, the decision is left to the LMs. This method also has limitations, such as being time-consuming and unstable.

## 4. Results

### 4.1. Datasets

The dataset used in the ISWC 2023 LM-KBC Challenge [19] is queried from Wikidata and further processed. It comprises 21 Wikidata relation types that cover 7 domains, including music, television series, sports, geography, chemistry, business, administrative divisions, and public figure information. It has 1,940 statements for each train, validation, and test sets. The results reported are based on the test set.<sup>7</sup> In the dataset, the minimum and maximum number of object-entities for each relation is different, ranging from 0 to 20. The minimum number of 0 means the subject-entities for some relations can have zero valid object-entities, for example, people still alive should not have a place or cause of death.

### 4.2. Model Performance

In terms of the overall performance of the model as shown in Table 1 and 3, GPT-4 is better than gpt-3.5-turbo. The retrieval-augmented context setting has the best performance compared with the other two few-shot learning settings. And the performance on question answering prompts and triple completion prompts is quite close.

---

<sup>7</sup>To investigate the actual knowledge gap between LLMs and Wikidata, we created ground truths of the test set through Wikidata SPARQL queries for offline evaluation. We report and analyze the offline evaluation results in Section 4 and the online evaluation results from CodaLab in Appendix A.

**Table 1**

Comparison of the performance of gpt-3.5-turbo and GPT-4 models based on the three settings: **question** prompting, **triple** completion prompting, and retrieval-augmented **context** setting. ‘baseline’ and ‘improved’ represent different disambiguation methods documented in Section 3.2.2. The best F1-scores among the three settings and two disambiguation methods of the models are highlighted.

Model	Disambiguation	question			triple			context		
		P	R	F1	P	R	F1	P	R	F1
gpt-3.5-turbo	baseline	0.557	0.574	0.540	0.545	0.579	0.525	0.599	0.659	0.593
	improved	0.581	0.597	0.563	0.576	0.609	0.554	0.625	0.684	<b>0.618</b>
gpt-4	baseline	0.650	0.661	0.632	0.641	0.651	0.624	0.650	0.685	0.641
	improved	0.682	0.689	0.661	0.678	0.683	0.657	0.676	0.709	<b>0.665</b>

From the lens of relations, as shown in the detailed results of GPT-4 (Table 2), LLMs perform well when the relation has a limited domain and/or range, for example, *PersonHasNobelPrize*, *CountryHasOfficialLanguage*, and *CompoundHasParts*. On the other hand, LLMs perform poorly for relations such as *PersonHasEmployer*, *PersonHasProfession*, and *PersonHasAutobiography*. This may be due to two reasons: firstly, LLMs have limited knowledge about public figures and their personal information (except for famous ones). Secondly, the unlimited answer space for such relations could increase the difficulty of prediction. The results show that LLMs perform relatively well on the knowledge of geography, as GPT-4 achieved F1-scores of 0.629 on *CityLocatedAtRiver*, 0.763 on *CountryBordersCountry*, 0.855 on *RiverBasinsCountry*, and 0.581 on *StateBordersState*, and the performance is inversely correlated with the size of the object range. The knowledge of public figures contained in LLMs could be an interesting topic to investigate since their performance across different aspects varies significantly. While LLMs correctly handle every instance of *PersonHasNobelPrize*, they also demonstrate relatively strong performance in areas such as place of birth and death, cause of death, and spouses. However, their performance tends to be deficient when it comes to details about individuals’ employers and professions.

### 4.3. Retrieval-Augmented Prediction

Providing relevant corpus as context to LLMs is an established method for improving model performance [31]. As such, we experimented with various sources and forms of context and selected the best ones for each relation. In particular, we experimented with using the introduction paragraphs of the Wikipedia article for the subject entity, the Infobox of the Wikipedia article for the subject entity in JSON format, as well as relation-specific sources such as IMDb. The effect of providing context varies for different models. It is observed gpt-3.5-turbo benefits from the context more compared with GPT-4. Reflected from F1-scores, the retrieval-augmented context setting exhibits an improvement of 0.055 compared with the question prompting setting for gpt-3.5-turbo and 0.004 for GPT-4.

In contrast to our intuition, adding context knowledge does not enhance the performance of GPT-4 in all relations as compared to only proving the few-shot examples, where only 10 out of 21 relations achieved better results in the context setting compared to the question and triple settings. Several factors may contribute to this, including the presence of a knowledge



**Table 2**

The results of probing GPT-4. For each relation, the improved disambiguation method used is listed, and the best F1-scores among the three settings are highlighted.

Relation	question			triple			context			Disambiguation
	P	R	F1	P	R	F1	P	R	F1	
BandHasMember	0.576	0.632	0.573	0.591	0.627	<b>0.581</b>	0.510	0.627	0.527	Keyword
CityLocatedAtRiver	0.780	0.562	0.615	0.775	0.578	<b>0.629</b>	0.648	0.504	0.533	LM
CompanyHasParentOrganisation	0.590	0.755	<b>0.590</b>	0.560	0.745	0.563	0.512	0.810	0.520	Baseline
CompoundHasParts	0.782	0.976	0.837	0.782	0.964	0.835	0.787	0.981	<b>0.843</b>	Case
CountryBordersCountry	0.802	0.685	0.730	0.806	0.688	0.734	0.829	0.723	<b>0.763</b>	Baseline
CountryHasOfficialLanguage	0.956	0.854	0.883	0.949	0.858	0.883	0.938	0.873	<b>0.886</b>	Keyword
CountryHasStates	0.796	0.809	0.800	0.754	0.748	0.750	0.805	0.816	<b>0.807</b>	LM
FootballerPlaysPosition	0.685	0.693	0.680	0.710	0.733	<b>0.708</b>	0.545	0.565	0.550	Case
PersonCauseOfDeath	0.765	0.783	0.762	0.795	0.803	0.793	0.800	0.803	<b>0.798</b>	Baseline
PersonHasAutobiography	0.478	0.471	<b>0.461</b>	0.458	0.486	<b>0.461</b>	0.475	0.471	0.459	Keyword
PersonHasEmployer	0.362	0.343	0.327	0.353	0.357	<b>0.328</b>	0.325	0.397	0.321	Case
PersonHasNobelPrize	1.000	1.000	<b>1.000</b>	1.000	1.000	<b>1.000</b>	1.000	1.000	<b>1.000</b>	Baseline
PersonHasNumberOfChildren	0.550	0.550	0.550	0.520	0.520	0.520	0.690	0.690	<b>0.690</b>	None
PersonHasPlaceOfDeath	0.670	0.730	0.670	0.690	0.730	0.690	0.783	0.810	<b>0.785</b>	Baseline
PersonHasProfession	0.494	0.420	0.427	0.538	0.422	<b>0.444</b>	0.390	0.408	0.363	Case
PersonHasSpouse	0.687	0.690	0.685	0.652	0.660	0.651	0.718	0.750	<b>0.727</b>	LM
PersonPlaysInstrument	0.566	0.565	0.531	0.559	0.519	0.507	0.559	0.597	<b>0.534</b>	Case
PersonSpeaksLanguage	0.747	0.813	0.744	0.755	0.836	<b>0.759</b>	0.757	0.808	0.742	Baseline
RiverBasinsCountry	0.841	0.946	<b>0.855</b>	0.841	0.931	0.852	0.827	0.941	0.852	Case
SeriesHasNumberOfEpisodes	0.590	0.590	0.590	0.530	0.530	0.530	0.690	0.690	<b>0.690</b>	None
StateBordersState	0.608	0.600	0.567	0.619	0.608	<b>0.581</b>	0.612	0.618	0.578	LM

gap and misaligned entity representations between Wikipedia and Wikidata. These factors could impact model performance, particularly when LLMs heavily rely on context enriched from Wikipedia. An example is *FootballerPlaysPosition*, where we have noted discrepancies between Wikipedia and Wikidata in the names used to represent identical or similar positions on the field. The investigation of this knowledge gap is explained in Section 5.2 and warrants further examination.

For most relations, where augmented context improved the performance, the introduction and Infobox of the Wikipedia page are sufficient based on the performance and the cost balance. Notable exceptions to the above are the *CountryHasState* and *SeriesHasNumberOfEpisodes* relations, where we augmented relation-specific context. For the *SeriesHasNumberOfEpisodes* relation, except for the previous two sources, we augmented the context from IMDb. The information on IMDb was added to the prompt prefaced by the label “IMDb”, and the model was asked to use this information (if it was available) to provide an answer. Moreover, for the *CountryHasState* relation, we discovered that GPT-4 would treat ‘state’ more like the definition of ‘country’ than that of the administrative division entity. Therefore, we experimented with providing the model with “Administrative Division of [entity]” Wikipedia page content, which outperformed the question setting for 0.007 of the F1-score.

#### 4.4. Disambiguation

When using the baseline disambiguation method, we observed disambiguation mistakes in 13 relations. These errors are categorized into two groups: **surface** disambiguation errors, in which the model produced the same strings of entities as the ground truths but assigned

incorrect QIDs, and **deep** disambiguation errors, where the model associated the same entities with different names (i.e., aliases) and also assigned incorrect QIDs. In this study, we focus only on addressing the former category while reserving discussion of the latter for future research. To tackle this challenge, we implemented improved disambiguation methods with the dual objective of rectifying errors to the fullest extent possible and concurrently reducing computational complexity.

From Table 1, we can observe an average increase in F1-scores of 0.0256 for all settings in the case of gpt-3.5-turbo and 0.0289 for GPT-4. For the 13 relations where improved disambiguation methods are applied, Table 2 listed the best-performing disambiguation method for each relation. Notably, for 3 relations (*CompoundHasParts*, *PersonPlaysInstrument*, and *RiverBasinsCountry*), the issues have been successfully solved. However, the rest 8 relations still remain either 2 or fewer unsolved errors, and 2 relations (*BandHasMember* and *StateBordersState*) face more than 7 unsolved errors, exceeding the capacity of their respective methods.

Given that the *wbsearchentities* Action API relies on label and alias-based searching, there’s a potential issue when LLMs predict objects with labels that are absent from the label and aliases of the corresponding Wikidata entity. This mismatch can lead to an incomplete list of candidate entities. From this perspective, LLMs have the ability to contribute to knowledge engineering by enriching the labels and aliases associated with Wikidata entities.

## 5. Discussion

### 5.1. Wikidata Quality

During the development of our pipeline and the evaluation of the results, it became apparent that the quality of Wikidata is an important issue, a problem that has also been discussed in previous works [32, 33]. For example, a large number of elements are missing for the relation *CompoundHasParts*, and many objects violate the value-type constraint of properties. In this situation, our proposed method would be useful for automatically providing suggestions and candidates for incomplete triples and thus enriching Wikidata by improving its quality. Moreover, it is possible to use LLMs to align the knowledge contained in Wikidata with the knowledge contained in Wikipedia and complete the triples of Wikidata using the Wikipedia articles as context. Furthermore, the performance of the LLMs on the object prediction task can be used as a metric to gauge the completeness of Wikidata entities. In cases where the difference between the predictions of the LLMs and the ground truth is substantial, the entity can be suggested to Wikidata editors for review using a recommender system, such as the one described by [34]. Finally, the labels (synonyms) of Wikidata entities are incomplete, which limits the ability of our disambiguation method since the system that retrieves the candidate entities needs labels and aliases to match the given string.

### 5.2. Knowledge Gap

Through our efforts to use Wikipedia as relevant context to improve the performance of LLMs in the object prediction task, we observed a significant knowledge gap between Wikipedia and Wikidata, which caused the performance of the model to deteriorate when provided with context



sourced from Wikipedia for some of the relations. To elucidate the cause of this phenomenon, we manually inspected several of these instances and realized that the information contained in Wikidata is different from the information contained in Wikipedia. One such example is the subject-relation pair *Ferrari S.p.A.*, *CompanyHasParentOrganisation*, for which LLMs correctly predicted the object *Exor*, matching the information on Wikipedia and the official report from Ferrari in 2021, whereas Wikidata contains the object *Ferrari N.V.*, which is outdated. This knowledge gap between Wikipedia and Wikidata is an open issue, and LLMs, either alone or by supporting human editors and suggesting edits, could play a pivotal role in addressing this issue and improving the data quality and recency of information contained in Wikidata. Finally, the knowledge gap is not limited to Wikidata and Wikipedia but appears to exist between LLMs as well. Specifically, as seen in Table 3, gpt-3.5-turbo outperforms the larger GPT-4 in two of the relations. Based on this, it stands to reason that different LLMs can contain different knowledge, and therefore, using an ensemble of LLMs with complementary strengths can lead to an improvement in performance.

## 6. Conclusion

Within the scope of the ISWC 2023 LM-KBC challenge, this work aimed at developing a method to probe LLMs for predicting the objects of Wikidata triples given the subject and relation. Our best-performing method achieved state-of-the-art results with a macro-averaged F1-score of 0.7007 across all relations, with GPT-4 having the best performance on the *PersonHasNobelPrize* relation and achieving a score of 1.0, while only achieving a score of 0.328 on the *PersonHasEmployer* relation. These results show that LLMs can be effectively used to complete knowledge bases when used in the appropriate context. At the same time, it is important to note that, largely due to the gaps in their knowledge, fully automatic knowledge engineering using LLMs is not currently possible for all domains, and a human-in-the-loop is still required to ensure the accuracy of the information.

## Acknowledgments

This work was partly funded by the HE project MuseIT, which has been co-founded by the European Union under the Grant Agreement No 101061441. Views and opinions expressed are, however, those of the authors and do not necessarily reflect those of the European Union or European Research Executive Agency.

## References

- [1] OpenAI, GPT-4 Technical Report, 2023. [arXiv:2303.08774](https://arxiv.org/abs/2303.08774).
- [2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, LLaMA: Open and Efficient Foundation Language Models, 2023. [arXiv:2302.13971](https://arxiv.org/abs/2302.13971).
- [3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu,

- J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023. [arXiv:2307.09288](https://arxiv.org/abs/2307.09288).
- [4] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, S. Singh, AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 4222–4235. URL: <https://aclanthology.org/2020.emnlp-main.346>. doi:10.18653/v1/2020.emnlp-main.346.
- [5] G. Qin, J. Eisner, Learning How to Ask: Querying LMs with Mixtures of Soft Prompts, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 5203–5212. URL: <https://aclanthology.org/2021.naacl-main.410>. doi:10.18653/v1/2021.naacl-main.410.
- [6] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, *ACM Comput. Surv.* 55 (2023). URL: <https://doi.org/10.1145/3560815>. doi:10.1145/3560815.
- [7] L. Ehrlinger, W. Wöß, Towards a Definition of Knowledge Graphs, *SEMANTICS (Posters, Demos, SuCESS)* 48 (2016) 2.
- [8] D. Fensel, U. Simsek, K. Angele, E. Huaman, E. Karle, O. Panasiuk, I. Toma, J. Umbrich, A. Wahler, *Knowledge Graphs: Methodology, Tools and Selected Use Cases*, 1st ed., Springer Publishing Company, Incorporated, 2020.
- [9] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. Gayo, R. Navigli, S. Neumaier, et al., *Knowledge Graphs, Synthesis Lectures on Data, Semantics, and Knowledge*, Morgan & Claypool Publishers, 2021. URL: <https://books.google.co.uk/books?id=hJ1NEAAAQBAJ>.
- [10] U. Simsek, E. Kärle, K. Angele, E. Huaman, J. Opendenplatz, D. Sommer, J. Umbrich, D. Fensel, A Knowledge Graph Perspective on Knowledge Engineering, *SN Comput. Sci.* 4 (2022). URL: <https://doi.org/10.1007/s42979-022-01429-x>. doi:10.1007/s42979-022-01429-x.
- [11] D. Vrandečić, M. Krötzsch, Wikidata: A Free Collaborative Knowledgebase, *Commun. ACM* 57 (2014) 78–85. URL: <https://doi.org/10.1145/2629489>. doi:10.1145/2629489.
- [12] A. Piscopo, E. Simperl, Who Models the World? Collaborative Ontology Creation and User Roles in Wikidata, *Proc. ACM Hum.-Comput. Interact.* 2 (2018). URL: <https://doi.org/10.1145/3274410>. doi:10.1145/3274410.
- [13] S. Razniewski, A. Yates, N. Kassner, G. Weikum, Language Models As or For Knowledge Bases, *CoRR* abs/2110.04888 (2021). URL: <https://arxiv.org/abs/2110.04888>. [arXiv:2110.04888](https://arxiv.org/abs/2110.04888).
- [14] D. Alivanistos, S. Santamaría, M. Cochez, J. Kalo, E. van Krieken, T. Thanapalasingam, Prompting as Probing: Using Language Models for Knowledge Base Construction, in: S. Singhanian, T.-P. Nguyen, S. Razniewski (Eds.), *LM-KBC 2022 Knowledge Base Construc-*

- tion from Pre-trained Language Models 2022, CEUR Workshop Proceedings, CEUR-WS.org, 2022, pp. 11–34.
- [15] B. Veseli, S. Singhania, S. Razniewski, G. Weikum, Evaluating Language Models For Knowledge Base Completion, in: *The Semantic Web: 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28–June 1, 2023, Proceedings*, Springer-Verlag, Berlin, Heidelberg, 2023, p. 227–243. URL: [https://doi.org/10.1007/978-3-031-33455-9\\_14](https://doi.org/10.1007/978-3-031-33455-9_14). doi:10.1007/978-3-031-33455-9\_14.
- [16] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying Large Language Models and Knowledge Graphs: A Roadmap, arXiv preprint arxiv:306.08302 (2023).
- [17] J. Wei, X. Wang, D. Schuurmans, M. Bosma, brian ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain of Thought Prompting Elicits Reasoning in Large Language Models, in: A. H. Oh, A. Agarwal, D. Belgrave, K. Cho (Eds.), *Advances in Neural Information Processing Systems*, 2022. URL: [https://openreview.net/forum?id=\\_VjQlMeSB\\_J](https://openreview.net/forum?id=_VjQlMeSB_J).
- [18] J. Huang, K. C.-C. Chang, Towards Reasoning in Large Language Models: A Survey, in: *Findings of the Association for Computational Linguistics: ACL 2023*, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 1049–1065. URL: <https://aclanthology.org/2023.findings-acl.67>. doi:10.18653/v1/2023.findings-acl.67.
- [19] S. Singhania, J.-C. Kalo, S. Razniewski, J. Z. Pan, LM-KBC: Knowledge base construction from pre-trained language models, *Semantic Web Challenge @ ISWC, CEUR-WS (2023)*. URL: <https://lm-kbc.github.io/challenge2023>.
- [20] Z. Zhong, D. Friedman, D. Chen, Factual Probing Is [MASK]: Learning vs. Learning to Recall, in: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Online, 2021, pp. 5017–5033. URL: <https://aclanthology.org/2021.naacl-main.398>. doi:10.18653/v1/2021.naacl-main.398.
- [21] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. Orr, L. Zheng, M. Yuksekogonul, M. Suzgun, N. Kim, N. Guha, N. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, Y. Koreeda, Holistic Evaluation of Language Models, 2022. arXiv:2211.09110.
- [22] H. Peng, X. Wang, S. Hu, H. Jin, L. Hou, J. Li, Z. Liu, Q. Liu, COPEN: Probing Conceptual Knowledge in Pre-trained Language Models, in: *Proceedings of EMNLP*, 2022.
- [23] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, A. Miller, Language Models as Knowledge Bases?, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2463–2473. URL: <https://aclanthology.org/D19-1250>. doi:10.18653/v1/D19-1250.
- [24] L. Yao, C. Mao, Y. Luo, KG-BERT: BERT for Knowledge Graph Completion, *CoRR abs/1909.03193* (2019). URL: <http://arxiv.org/abs/1909.03193>. arXiv:1909.03193.
- [25] A. Rogers, M. Gardner, I. Augenstein, QA Dataset Explosion: A Taxonomy of NLP Resources for Question Answering and Reading Comprehension, *ACM Comput. Surv.* 55 (2023). URL:

<https://doi.org/10.1145/3560260>. doi:10.1145/3560260.

- [26] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, E. Simperl, T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), European Language Resources Association (ELRA), Miyazaki, Japan, 2018. URL: <https://aclanthology.org/L18-1544>.
- [27] R. Speer, C. Havasi, Representing General Relational Knowledge in ConceptNet 5, in: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), European Language Resources Association (ELRA), Istanbul, Turkey, 2012, pp. 3679–3686. URL: [http://www.lrec-conf.org/proceedings/lrec2012/pdf/1072\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2012/pdf/1072_Paper.pdf).
- [28] J.-C. Kalo, L. Fichtel, KAMEL: Knowledge Analysis with Multitoken Entities in Language Models, in: Automated Knowledge Base Construction, 2022.
- [29] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard, et al., KILT: a Benchmark for Knowledge Intensive Language Tasks, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 2523–2544.
- [30] J. Yu, X. Wang, S. Tu, S. Cao, D. Zhang-Li, X. Lv, H. Peng, Z. Yao, X. Zhang, H. Li, C. Li, Z. Zhang, Y. Bai, Y. Liu, A. Xin, N. Lin, K. Yun, L. Gong, J. Chen, Z. Wu, Y. Qi, W. Li, Y. Guan, K. Zeng, J. Qi, H. Jin, J. Liu, Y. Gu, Y. Yao, N. Ding, L. Hou, Z. Liu, B. Xu, J. Tang, J. Li, KoLA: Carefully Benchmarking World Knowledge of Large Language Models, 2023. arXiv:2306.09296.
- [31] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf).
- [32] A. Piscopo, E. Simperl, What we talk about when we talk about wikidata quality: a literature survey, in: B. Lundell, J. Gamalielsson, L. Morgan, G. Robles (Eds.), Proceedings of the 15th International Symposium on Open Collaboration, OpenSym 2019, Skövde, Sweden, August 20-22, 2019, ACM, 2019, pp. 17:1–17:11. URL: <https://doi.org/10.1145/3306446.3340822>. doi:10.1145/3306446.3340822.
- [33] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe, P. A. Szekely, A Study of the Quality of Wikidata, *J. Web Semant.* 72 (2022) 100679. URL: <https://doi.org/10.1016/j.websem.2021.100679>. doi:10.1016/j.websem.2021.100679.
- [34] K. Alghamdi, M. Shi, E. Simperl, Learning to Recommend Items to Wikidata Editors, in: A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P. M. Barnaghi, A. Haller, M. Dragoni, H. Alani (Eds.), The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings, volume 12922 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 163–181. URL: [https://doi.org/10.1007/978-3-030-88361-4\\_10](https://doi.org/10.1007/978-3-030-88361-4_10). doi:10.1007/978-3-030-88361-4\_10.

## A. Online Evaluation Results

**Table 3**

The online evaluation results from CodaLab. The results are aggregated from the highest ones in the three settings for each relation and model. The online evaluation results from CodaLab. The results are aggregated from the highest ones in the three settings for each relation and model. The outcomes conducted in online evaluation have demonstrated superior results compared to the offline ones, with particular significance observed in the cases of *CompoundHasParts* and *CityLocatedAtRiver*. This phenomenon can be attributed to the revision of online ground truths. Additionally, this observation emphasizes that LLMs have the potential to enhance the overall quality of Wikidata.

Relation	gpt-3.5-turbo				GPT-4			
	P	R	F1	Setting	P	R	F1	Setting
BandHasMember	0.5378	0.5830	0.5295	triple	0.5905	0.6331	<b>0.5838</b>	triple
CityLocatedAtRiver	0.5500	0.4723	0.4845	context	0.7600	0.6538	<b>0.6792</b>	triple
CompanyHasParentOrganisation	0.4300	0.7500	0.4267	context	0.6100	0.7650	<b>0.6100</b>	question
CompoundHasParts	0.9591	0.9659	0.9615	context	0.9962	1.0000	<b>0.9978</b>	context
CountryBordersCountry	0.8628	0.7756	<b>0.8107</b>	context	0.8292	0.7699	0.7937	context
CountryHasOfficialLanguage	0.9313	0.8731	0.8814	question	0.9379	0.8821	<b>0.8932</b>	context
CountryHasStates	0.7926	0.7772	0.7823	context	0.8048	0.8156	<b>0.8073</b>	context
FootballerPlaysPosition	0.6400	0.6333	0.6323	triple	0.7100	0.7333	<b>0.7083</b>	triple
PersonCauseOfDeath	0.7600	0.7833	0.7550	question	0.8000	0.8033	<b>0.7983</b>	context
PersonHasAutobiography	0.4337	0.5000	0.4490	context	0.4483	0.4850	<b>0.4583</b>	context
PersonHasEmployer	0.3053	0.4087	0.3134	context	0.3533	0.3567	<b>0.3282</b>	triple
PersonHasNoblePrize	0.9900	0.9900	0.9900	question	1.0000	1.0000	<b>1.0000</b>	question
PersonHasNumberOfChildren	0.6900	0.6900	0.6900	context	0.7000	0.7000	<b>0.7000</b>	context
PersonHasPlaceOfDeath	0.6150	0.7800	0.6167	context	0.7833	0.8100	<b>0.7850</b>	context
PersonHasProfession	0.2875	0.3927	0.3029	context	0.5375	0.4159	<b>0.4395</b>	triple
PersonHasSpouse	0.7583	0.7850	<b>0.7650</b>	context	0.7083	0.7450	0.7183	context
PersonPlaysInstrument	0.3987	0.4946	0.4087	context	0.5485	0.5924	<b>0.5279</b>	context
PersonSpeaksLanguage	0.8683	0.6893	0.7344	triple	0.7550	0.8360	<b>0.7589</b>	triple
RiverBasinsCountry	0.7869	0.8986	0.8054	context	0.8408	0.9463	<b>0.8549</b>	question
SeriesHasNumberOfEpisodes	0.6200	0.6300	0.6233	context	0.6900	0.6900	<b>0.6900</b>	context
StateBordersState	0.5753	0.5898	0.5435	context	0.6139	0.6135	<b>0.5811</b>	triple
Zero-object cases	0.4708	0.7559	0.5802	/	0.5026	0.9202	<b>0.6501</b>	/
Average	0.6568	0.6887	0.6432	/	0.7151	0.7260	<b>0.7007</b>	/

## B. Prompt Templates

### BandHasMember

Who are the members of {subject\_entity}? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### CityLocatedAtRiver

Which river is {subject\_entity} located at? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### CompanyHasParentOrganisation

{subject\_entity} is a subsidiary of which company? Return a Python list with an empty string

(i.e. [""]) if none. Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **CountryBordersCountry**

Which countries share borders with {subject\_entity}? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **CountryHasOfficialLanguage**

What is the official language of {subject\_entity}? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **CountryHasStates**

What are the first-level administrative territorial entities of {subject\_entity}? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **FootballerPlaysPosition**

What position does {subject\_entity} play in football? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **PersonCauseOfDeath**

What caused the death of {subject\_entity}? If none or still alive, return [""]. Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **PersonHasAutobiography**

What is the title of {subject\_entity}'s autobiography? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **PersonHasEmployer**

Who is {subject\_entity}'s employer? Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **PersonHasNoblePrize**

Which Nobel Prize did {subject\_entity} receive? Select from this list: ["Nobel Peace Prize", "Nobel Prize in Literature", "Nobel Prize in Physics", "Nobel Prize in Chemistry", "Nobel Prize in Physiology or Medicine"]. Return a Python list with an empty string (i.e. [""]) if none. Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **PersonHasNumberOfChildren**

How many children does {subject\_entity} have? Return the string format of the number only. Format the response as a Python list such as ["answer\_a", "answer\_b"].

### **PersonHasPlaceOfDeath**

Where did {subject\_entity} die? Return a Python list with an empty string (i.e. [""]) if he or she is still alive. Format the response as a Python list such as ["answer\_a", "answer\_b"].



**PersonHasProfession**

What is {subject\_entity}'s profession or occupation? Format the response as a Python list such as ["answer\_a", "answer\_b"].

**PersonHasSpouse**

What is the name of the spouse of {subject\_entity}? Format the response as a Python list such as ["answer\_a", "answer\_b"].

**PersonPlaysInstrument**

What instruments does {subject\_entity} play? Format the response as a Python list such as ["answer\_a", "answer\_b"].

**PersonSpeaksLanguage**

What languages does {subject\_entity} speak? Format the response as a Python list such as ["answer\_a", "answer\_b"].

**RiverBasinsCountry**

In which country can you find the {subject\_entity} river basin? Format the response as a Python list such as ["answer\_a", "answer\_b"].

**SeriesHasNumberOfEpisodes**

How many episodes does the series {subject\_entity} have? Return the string format of the number. Format the response as a Python list such as ["answer\_a", "answer\_b"].

**CompoundHasParts**

What are the chemical components of {subject\_entity}? Return the full name of components such as ["carbon", "nitrogen"]. Format the response as a Python list such as ["answer\_a", "answer\_b"].

**StateBordersState**

Which states border the state of {subject\_entity}? Format the response as a Python list such as ["answer\_a", "answer\_b"].