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

Bohui Zhang[1], Ioannis Reklos[1], Nitisha Jain[1], Albert Meroño Peñuela[1] and Elena Simperl[1]

[1]*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 0.689 (0.7007 online evaluation) 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 Bases (e.g., Wikidata) completion and correction. The investigation of the results also suggests the potential contribution of LLMs in collaborative KE. The code 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 ChatGPT [1] and GPT-4 models [2], which induced the development of several other LLMs such as LLama from Meta [3], Claude from Anthropic[1], and Bard from Alphabet[2].

This surge in the development and release of large LMs, many of which have been trained with Reinforcement Learning with Human Feedback (RLHF) [4], 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

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. Reklos); 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)

[1]https://claude.ai/

[2]https://bard.google.com/

past [5, 6].

The idea of using LMs to construct and complete knowledge graphs (KGs) has been investigated by many studies [4, 7, 8]. However, the recent boost in performance has once again brought to the surface the question of using LMs for, or even as, KGs.

Despite the incredible promise of LLMs as knowledge stores, there are fundamental differences that set them apart and even disadvantage them as compared to KGs. The reasoning and inference power of the KGs are the most important of these differences. 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. Additionally, the most popular 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 while being largely ignorant of the lesser-known ones.

In this paper, we describe our approach to using LLMs for Knowledge Engineering (KE) tasks, especially targeting solving the ISWC 2023 LM-KBC Challenge, and report our findings regarding the prospect of using these models to automate the process of KE. 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* (`Q234021`). We used two state-of-the-art LLMs, gpt-3.5-turbo[3] and GPT-4 [2] for this task. By performing different experiments using few-shot approaches, as well as leveraging appropriate context, we have been able to achieve a macro-average F1 score of 0.689 (0.7007 on CodaLab), 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. In particular, several previous works have attempted to use language models to construct or complete knowledge graphs. Among early works, the LAMA paper by Petroni et al. [9, 10] investigated the task of knowledge graph completion with language models by extracting Wikidata facts via probing the LMs. Along similar lines, KG-BERT leverages the BERT language model to perform the link prediction task for knowledge graph completion[11]. The extent of the usefulness of LLMs for the construction and completion of knowledge graphs has since been further analyzed [12]. Follow up work after the LAMA improved the performance even further [5, 13]. Prompt engineering has caught the attention of many recent works that aim to elicit knowledge from the language models [7, 14]. These works are the most similar to our approach in this paper.

[3] https://platform.openai.com/docs/models/gpt-3-5

2.2. Benchmarks and Datasets

To fulfil the need for investigating the ability of LLMs to perform knowledge-intensive tasks, knowledge-oriented benchmarks and datasets have been proposed. They can be primarily classified as two types: question answering and fact completion [15]. In the fact completion datasets, facts are formatted in (subject, relation, object(s)) triples. As a very first dataset in the LM era, LAMA was constructed from a variety of knowledge graph sources of factual and commonsense knowledge. KAMEL [16] extended LAMA from single token objects to multi-token. KILT [17] was constructed from Wikipedia pages and can be applied to various knowledge-intensive language tasks. WikiFact [15] constructed as part of the HELM benchmark covers broader domains, and, finally, KoLA [18] aimed at measuring the real-world performance of LLMs by expanding beyond language modelling and attempting to measure the ability of the models in all facets of knowledge processing, ranging from knowledge memorization to knowledge creation.

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 $\{w_{o_i}, \cdots, w_{o_n}\}$.

3.2. LLM-based Knowledge Engineering (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 [2] 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. Providing all potential answers at the prompt is likely to help LLMs perform well (F1-score close to 1) and return the desired format.

In the third setting, we provide context to help LLMs by enriching knowledge. In the first step, we ask LLMs to predict the objects based on their own knowledge using the same settings as question prompting. Then we provided the context, and we let LLMs predict again by considering the context and comparing it with their own predictions. In this study, we used

Wikipedia as the general context corpus. 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. LLMs were asked to make predictions again by considering the context and comparing it with the previous response.

In all settings, we perform few-shot learning, where we first provide three examples. Since the required format of results is a list, providing examples with the exact format is expected to help LLMs return better-formatted results. In the dataset, there are some relations that could potentially have empty results. In this case, the prompt indicated the required return format (i.e., [“”]).

3.2.2. Wikidata Entity Mapping

The entity mapping step first finds Wikidata entities for each object string using the MediaWiki Action API[4]. One of the actions, *wbsearchentities* 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. To reduce the cost while improving the accuracy for disambiguation, we treated different relations with three methods: *case-based*, *keyword-based*, and *LM-based*.

The *case-based* method is a hard-coding solution for efficiently solving ambiguities for relations with smaller answer spaces and limited corner cases. For example, *CompoundHasParts* only has all the chemical elements as its answer space. Further, it only has one ambiguous case: ‘*mercury*’. 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 entity 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 and descriptions, 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]https://www.wikidata.org/w/api.php

Table 1
Comparison of the performance of gpt-3.5-turbo and GPT-4 models.

Model	question			triple			context		
	P	R	F1	P	R	F1	P	R	F1
gpt-3.5-turbo	0.581	0.597	0.563	0.576	0.609	0.554	0.625	0.684	0.618
GPT-4	0.682	0.689	0.661	0.678	0.683	0.657	0.676	0.709	**0.665**

4. Results

4.1. Datasets

The dataset used in the ISWC 2023 LM-KBC Challenge[5] 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.[6] 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.

4.2. Model Performance

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

From the lens of relations, 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 well on the knowledge of geography (*CityLocatedAtRiver*, *CountryBordersCountry*, *CountryHasStates*, *RiverBasinsCountry*, *StateBordersState*), and the performance is inversely correlated with the size of the object range.

4.3. In-context Learning

Providing context to LLMs is an established method for improving model performance [10**?**]. As such, we experimented with various sources and forms of context and selected the best one for each relation. In particular, we experimented with using the introduction content of the Wikipedia article for the subject entity, the Infobox of the Wikipedia article for the subject

[5]https://github.com/lm-kbc/dataset2023

[6]To investigate the actual knowledge gap between LLMs and Wikidata, we created ground truths of the test set for offline evaluation. The online evaluation results from CodaLab are reported in Appendix A.

Table 2
The results of probing GPT-4 with few-shot examples. The 'context' represents question prompts with Wikipedia context. All results have been disambiguated. For each relation, the best F1-scores among the three settings are highlighted.

Relation	question			triple			context		
	P	R	F1	P	R	F1	P	R	F1
BandHasMember	0.576	0.632	0.573	0.591	0.627	**0.581**	0.510	0.627	0.527
CityLocatedAtRiver	0.780	0.562	0.615	0.775	0.578	0.629	0.648	0.504	0.533
CompanyHasParentOrganisation	0.590	0.755	**0.590**	0.560	0.745	0.563	0.512	0.810	0.520
CompoundHasParts	0.782	0.976	0.837	0.782	0.964	0.835	0.787	0.981	**0.843**
CountryBordersCountry	0.802	0.685	0.730	0.806	0.688	0.734	0.829	0.723	**0.763**
CountryHasOfficialLanguage	0.956	0.854	0.883	0.949	0.858	0.883	0.938	0.873	**0.886**
CountryHasStates	0.796	0.809	0.800	0.754	0.748	0.750	0.805	0.816	**0.807**
FootballerPlaysPosition	0.685	0.693	0.680	0.710	0.733	**0.708**	0.545	0.565	0.550
PersonCauseOfDeath	0.765	0.783	0.762	0.795	0.803	0.793	0.800	0.803	**0.798**
PersonHasAutobiography	0.478	0.471	**0.461**	0.458	0.486	**0.461**	0.475	0.471	0.459
PersonHasEmployer	0.362	0.343	0.327	0.353	0.357	**0.328**	0.325	0.397	0.321
PersonHasNobelPrize	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PersonHasNumberOfChildren	0.550	0.550	0.550	0.520	0.520	0.520	0.690	0.690	**0.690**
PersonHasPlaceOfDeath	0.670	0.730	0.670	0.690	0.730	0.690	0.783	0.810	**0.785**
PersonHasProfession	0.494	0.420	0.427	0.538	0.422	**0.444**	0.390	0.408	0.363
PersonHasSpouse	0.687	0.690	0.685	0.652	0.660	0.651	0.718	0.750	**0.727**
PersonPlaysInstrument	0.566	0.565	0.531	0.559	0.519	0.507	0.559	0.597	**0.534**
PersonSpeaksLanguage	0.747	0.813	0.744	0.755	0.836	**0.759**	0.757	0.808	0.742
RiverBasinsCountry	0.841	0.946	**0.855**	0.841	0.931	0.852	0.827	0.941	0.852
SeriesHasNumberOfEpisodes	0.590	0.590	0.590	0.530	0.530	0.530	0.690	0.690	**0.690**
StateBordersState	0.608	0.600	0.567	0.619	0.608	**0.581**	0.612	0.618	0.578
Average	0.682	0.689	0.661	0.678	0.683	0.657	0.676	0.709	**0.665**

entity in JSON format, as well as relation-specific sources of information 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. In contrast to our intuition, adding context did not improve the performance of GPT-4 in all relations as compared to the few-shot setting. For most properties, where context improved the performance, the introduction and Infobox of the Wikipedia page are sufficient. Notable exceptions to the above are the *SeriesHasNumberOfEpisodes* and the *CountryHasState* relations. For the *SeriesHasNumberOfEpisodes* relation, we augmented the Wikipedia-based context with context provided 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 different contexts and realized that the model provided the most accurate results when provided with the Wikipedia page for the term "Administrative Division". More information on the experimental setting for each relation can be seen in Table 3.

Table 3
The context types and disambiguation methods used for each relation.

Relation	Context type	Disambiguation method
BandHasMember	Wikipedia Intro + Infobox	Keyword-based
CityLocatedAtRiver	Wikipedia Intro + Infobox	LM-based
CompanyHasParentOrganisation	Wikipedia Intro + Infobox	-
CompoundHasParts	Wikipedia Intro + Infobox	Case-based
CountryBordersCountry	Wikipedia Intro + Infobox	-
CountryHasOfficialLanguage	Wikipedia Intro + Infobox	Keyword-based
CountryHasStates	Wikipedia Page	LM-based
FootballerPlaysPosition	Wikipedia Intro + Infobox	Case-based
PersonCauseOfDeath	Wikipedia Intro + Infobox	-
PersonHasAutobiography	Wikipedia Intro + Infobox	Keyword-based
PersonHasEmployer	Wikipedia Intro + Infobox	Case-based
PersonHasNobelPrize	Wikipedia Intro + Infobox	-
PersonHasNumberOfChildren	Wikipedia Intro + Infobox	-
PersonHasPlaceOfDeath	Wikipedia Intro + Infobox	-
PersonHasProfession	Wikipedia Intro + Infobox	Case-based
PersonHasSpouse	Wikipedia Intro + Infobox	LM-based
PersonPlaysInstrument	Wikipedia Intro + Infobox	Case-based
PersonSpeaksLanguage	Wikipedia Intro + Infobox	-
RiverBasinsCountry	Wikipedia Intro + Infobox	Case-based
SeriesHasNumberOfEpisodes	IMDb + Wikipedia Intro + Infobox	-
StateBordersState	Wikipedia Intro + Infobox	LM-based

4.4. Disambiguation

When employing the baseline disambiguation method provided by the challenge, we noticed ambiguities for 13 relations in total, with the model predicting the correct string but the returned QID being different from the ground truth. To remedy this issue, we employed different disambiguation methods with increasing computational costs. Specifically, we experimented with baseline Wikidata-based, keyword-based, case-based, and LM-based disambiguation methods. The best-performing disambiguation method for each relation is shown in Table **??**. From Table 4, we can observe that F1-score increases for all settings and models.

Table 4
The results of disambiguation methods.

Model	Setting	Baseline			Disambiguation		
		P	R	F1	P	R	F1
gpt-3.5-turbo	question	0.557	0.574	0.540	0.581	0.597	**0.563**
	triple	0.545	0.579	0.525	0.576	0.609	**0.554**
	question (context)	0.599	0.659	0.593	0.625	0.684	**0.618**
GPT-4	question	0.650	0.661	0.632	0.682	0.689	**0.661**
	triple	0.641	0.651	0.624	0.678	0.683	**0.657**
	question (context)	0.650	0.685	0.641	0.676	0.709	**0.665**

5. Discussion

5.1. Wikidata Quality

During the construction of datasets and evaluation, it became apparent that the quality of Wikidata is an important issue, a problem that has also been discussed in previous works [19, 20]. 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 imperfect statements 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 by using LLMs to complete 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 the editors for review using a recommender system, such as the one described by [21]. 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 task, we observed a significant knowledge gap between Wikipedia and Wikidata, which caused the performance of the model for some of the relations to deteriorate when provided with context sourced from Wikipedia. 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 pair *Ferrari S.p.A., CompanyHasParentOrganisation*, for which LLMs 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.*. 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.

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 property. Our best-performing method achieved state-of-the-art results with a macro-averaged F1-score of 0.689 (0.7007 online evaluation) across all properties, 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.

References

[1] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al., A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, arXiv preprint arXiv:2302.04023 (2023).

[2] OpenAI, GPT-4 Technical Report, 2023. arXiv:2303.08774.

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., LLaMA: Open and Efficient Foundation Language Models, arXiv preprint arXiv:2302.13971 (2023).

[4] Z. Li, Z. Yang, M. Wang, Reinforcement Learning with Human Feedback: Learning Dynamic Choices via Pessimism, 2023. arXiv:2305.18438.

[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] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, S. Singh, AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts, arXiv preprint arXiv:2010.15980 (2020).

[7] D. Alivanistos, S. B. Santamaría, M. Cochez, J. C. Kalo, E. van Krieken, T. Thanapalasingam, Prompting as Probing: Using Language Models for Knowledge Base Construction, in: 2022 Semantic Web Challenge on Knowledge Base Construction from Pre-Trained Language Models, LM-KBC 2022, CEUR-WS. org, 2022, pp. 11–34.

[8] S. Singhania, T.-P. Nguyen, S. Razniewski, LM-KBC: Knowledge base construction from pre-trained language models, 2022.

[9] F. Petroni, T. Rocktäschel, P. S. H. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, S. Riedel, Language Models as Knowledge Bases?, CoRR abs/1909.01066 (2019). URL: http://arxiv.org/abs/1909.01066. arXiv:1909.01066.

[10] F. Petroni, P. S. H. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, S. Riedel, How Context Affects Language Models' Factual Predictions, in: D. Das, H. Hajishirzi, A. McCallum, S. Singh (Eds.), Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020, 2020. URL: https://doi.org/10.24432/C5201W. doi:10.24432/C5201W.

[11] 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.

[12] S. Razniewski, A. Yates, N. Kassner, G. Weikum, Language models as or for knowledge bases, arXiv preprint arXiv:2110.04888 (2021).

[13] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li,

X. V. Lin, et al., OPT: Open Pre-trained Transformer Language Models, arXiv preprint arXiv:2205.01068 (2022).

[14] T. Li, W. Huang, N. Papasarantopoulos, P. Vougiouklis, J. Z. Pan, Task-specific Pre-training and Prompt Decomposition for Knowledge Graph Population with Language Models (2022).

[15] 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. Yuksekgonul, 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.

[16] J.-C. Kalo, L. Fichtel, KAMEL: Knowledge Analysis with Multitoken Entities in Language Models, in: Automated Knowledge Base Construction, 2022.

[17] 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.

[18] 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, CoRR abs/2306.09296 (2023). URL: https://doi.org/10.48550/arXiv.2306.09296. doi:10.48550/arXiv.2306.09296. arXiv:2306.09296.

[19] 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.

[20] 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.

[21] 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. doi:10.1007/978-3-030-88361-4_10.

A. Supplementary Evaluation Results

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

Table 5
The online evaluation results from CodaLab. The results are aggregated from the highest ones in the three settings for each relation and model.