Noise or Nuance: An Investigation Into Useful Information and Filtering For LLM Driven AKBC

Alex Clay[1,*], Ernesto Jiménez-Ruiz[1] and Pranava Madhyastha[1,2]

[1]City St George's, University of London, Northampton Square, London, EC1V 0HB, United Kingdom
[2]The Alan Turing Institute, British Library, 96 Euston Rd., London NW1 2DB, United Kingdom

Abstract

RAG and fine-tuning are prevalent strategies for improving the quality of LLM outputs. However, in constrained situations, such as that of the 2025 LM-KBC challenge, such techniques are restricted. In this work we investigate three facets of the triple completion task: generation, quality assurance, and LLM response parsing. Our work finds that in this constrained setting: additional information improves generation quality, LLMs can be effective at filtering poor quality triples, and the tradeoff between flexibility and consistency with LLM response parsing is setting dependent.

1. Introduction

Retrieval Augmented Generation is often implemented to improve the performance of LLMs on a variety of tasks, including those pertaining to knowledge extraction [1, 2]. Automated Knowledge Base Completion (AKBC) has long been a topic of interest [3], and has more recently seen an increase in investigations relating to the use of LLMs, not only to create KBs [4, 5], but in some instances to act in place of them. We adopt one such setting and task, introduced by the 2025 LM-KBC challenge, to utilise only the given data and Qwen3's [6] 8 Billion parameter variation without finetuning or RAG to generate completions for KG triples[1]. This provided the opportunity to investigate specific facets of what aspects of generation improve triple completion under such constraints. Therefore, our task was to generate high quality tail entities to complete a triple, without augmentation to the assigned LLM. We considered means of emulating aspects of RAG and finetuning without implementing either.

The presence of grounding information when prompting LLMs can influence the quality of the generated output [7]. As such, even LLM generated supporting information might be able to improve efficacy of example-supported generation in situations with sparse external data. Moreover, without external data or a ground truth, evaluation of triple quality lacks a key point of comparison. LLMs have been utilised as judges [8], which provide a convenient means of checking generated outputs against the model's own knowledge. Additionally, handling and formatting of LLM responses are crucial to avoiding error propagation; while often convenient to use NLP libraries like SpaCy[2] for parsing, the LLM could potentially complete the task itself. In our work, we hypothesized that it is possible to utilise the LLM in a number of ways to holistically replace augmentation like finetuning and external knowledge sources through handling: grounding information for generation, filtration of poor quality triples, and cleaning of LLM responses. As such, we present an investigation of the following questions within the constrained setting of the LM-KBC challenge:

RQ1: What type of information is useful to generating candidate tails with limited access to external data?
RQ2: Without access to the ground truth, what is an effective means for filtering poor quality triples?
RQ3: Is the flexibility of LLM entity extraction worth the tradeoff of inconsistency?

Joint proceedings of KBC-LM and LM-KBC @ ISWC 2025
*Corresponding author.
alex.clay@city.ac.uk (A. Clay); ernesto.jimenez-ruiz@city.ac.uk (E. Jiménez-Ruiz); pranava.madhyastha@city.ac.uk (P. Madhyastha)
0009-0004-2237-7412 (A. Clay); 0000-0002-9083-4599 (E. Jiménez-Ruiz); 0000-0002-4438-8161 (P. Madhyastha)

[1]Our code: https://github.com/alexclay42/lmkbcWorkshop
[2]https://spacy.io

Our primary findings indicate that within our constrained setting: additional information improves generation quality, LLMs are effective at filtering poor quality triples if not a bit strict, and the tradeoff between flexibility and consistency in LLM response parsing varies based on the setting.

2. Related Work

In recent years, LLMs have been more commonly used for the task of automatic knowledge base completion. Hu et al. [4], for instance, employed LLM-generated information about a given topic in the form of triples to retrieve new entities, applying the same technique to the newly returned entities. He et al. [9] meanwhile, focused on a setting with few available example triples, and determined that few-shot examples can improve one and two-hop relations. Cohen et al. [5] created a knowledge graph from a seed entity through the generation of potential relations and subsequent tails. The creation of the knowledge graph relied entirely on the knowledge of the chosen LLM to provide candidates in this approach, introducing paraphrasing of the entities during the relation generation and relations during tail generation to increase quantity and diversity of the candidates. In our work, we adapted the paraphrasing concept to be a brief factual explanation of the entity and used the same format as [5] for the examples provided at generation.

Masked Language models, such as BERT, have also been employed for this task. For example, in the original LAMA Probe, Petroni et al. [10] investigated BERT for answering fill-in-the-blank style queries. Jiang et al. [11] improved the SOTA on the LAMA task by over 8 percent by utilising more effective prompts through mining and paraphrasing-based methods. Additionally Veseli et al. [12] found that BERT had good generalization ability and that prompt formulation and vocabulary expansion could significantly improve its capability. Though in a different format, we apply a similar concept of eliciting the LLM to fill in the blank, of completing the given format rather than a sentence.

In order to derive understanding of LLMs as knowledge bases, many have investigated the information inherently possessed in an LLM's parameters. Hu et al. [4] noted that availability bias is a crucial issue as, essentially, only surface information is being reached, and that which is beyond the benchmarking or user-induced information is not being adequately explored. Moreover, LLMs can be sensitive to perturbations as the naming of entities can impact 'model cognition' [13] and that LLMs are not terribly adept at relating new knowledge with knowledge they already possess. Moreover, Berglund et al. [14] found that LLMs typically do not learn to reverse relationships unless something like "A is B" appears in-context. When utilising LLMs, the information presented with and the prompt itself greatly influence the generated response. Chen et al. [15] indicated the importance of model-specific prompt adaptation, something avoided by Wu et al. [16]'s promptless knowledge estimator where they instead utilise similarly related entity and tail pairs. We utilise a similar promptless strategy in our approach, to more directly assess the impact of different information in eliciting generation of tail candidates.

3. Methodology

Our task, as with the LM-KBC challenge[3], was to complete a set of given initial entity and relation pairs, and to elicit the designated LLM, Qwen3 8B, to provide a correct tail entity for each pair. A triple is the underlying structure in a knowledge graph with the format of two entities with a relation between them, the later of which is a 'tail' entity. Some of the entity/relation pairs might have one, multiple, or no actual tail entity. Our research questions address three crucial aspects of this task generation, evaluation, and parsing; and how to successfully address it within the provided limitations. To address the task, we constructed a three step program: firstly an initial generation, then re-generation, and completing with a filtration process. We present an illustration in Figure 1 showing the linear process and filtration variations.

We follow a general approach to generate an initial set of tail entities for the dataset, using the same example based prompt as Cohen et al. [5]. This initial set of candidates could then be sent to the

[3]https://lm-kbc.github.io/challenge2025/

re-generation component based on the experiment. The re-generation component was introduced to provide an opportunity for poor quality triples to be improved with the intention of reducing the number of candidates removed during filtering. Each triple was evaluated with a call to the LLM to judge its quality and regenerated on failure. This process could happen up to 3 times, if the candidate triple was continually rejected. The candidate triples then reached the filtration step, and either encountered the judge, consensus, or translation approach before formatting for evaluation. Following each component, the LLM response was cleaned into a candidate tail before proceeding (ie. 'The answer is Isla Nublar' becomes 'Isla Nublar'), a process which itself provided means for comparing cleaning techniques. Each of these components contributes to both the completion of the task and investigation of our research questions.

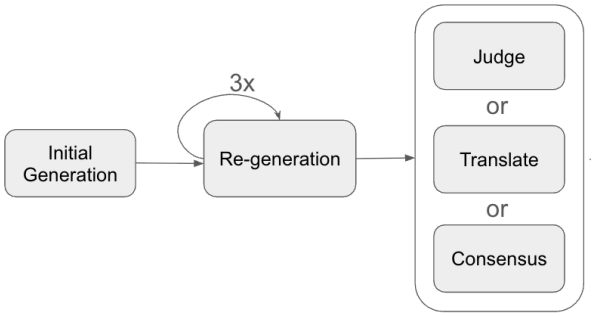

Figure 1: The component flow for the experiments.

RQ1 In order to address the question of what information is useful to generation (RQ1), we modified the examples and additional information at the initial and re-generation points. Firstly, we investigated the introduction of LLM-generated supporting information about the target entity, gained through prompting the LLM to return facts about it. This information was provided in combination with the examples at initial generation and re-generation to determine if it could act similarly to RAG grounding in improving the efficacy. Another type of information we investigated was the addition of a guide sentence to indicate the expected output type (such as a name or country) in place of the LLM generated grounding. Many approaches, such as the baseline approach for the challenge[4], utilised relation-specific methods, in that each different type of relation had a specific type of prompt. We adapted this concept into this guide sentence to determine if it aided the LLM in producing the correct output. We also investigated variation in the example triples provided, comparing that of relation-specific and entity/relation pair specific examples. The relation specific examples were the same relation type as the target entity/relation pair, whereas the specific examples were selected on a pair by pair basis through using the LLM to select the three most similar examples of the same relation type from the LM-KBC validation dataset. The intention was to determine if having more similar examples might ground the generation in more similar information to that of the target, such as potentially returning examples with other islands if the target entity was an island. We aimed for the information grounding and specific examples to potentially act as a simulacra of RAG, and for the relation specific guides and previous errors to emulate fine tuning.

RQ2 Due to the constrained setting, it was necessary to implement a means of quality control that did not rely on external verification, the focus of RQ2. Inspired by Wiland et al. [17], who used an approach which leveraged the LLM's ability to predict the log-likelihood of a statement to rank answer options, we utilised the LLM to indicate the quality of the triples. We compared three variations of filtration based off of different concepts: LLM as a judge [8], consensus similar to the duplicate reduction and minimum instances in Cohen et al. [5], and a comparison of translation between formats. For the judge

[4]https://github.com/lm-kbc/dataset2025

filter, the LLM was prompted to return a value between 0 and 100 indicating the quality of the triple. This was done 3 times, and if the average score was 50 or more, the candidate triple was accepted. We use the same judge for the re-generation and filter, though increase the calls to 3 times and take the average, to better handle nonsensical answers and increase confidence of the ruling. The consensus filter required any predicted tail to appear twice or more to be accepted, as more frequent appearances might indicate higher confidence in a candidate tail. For the translation filter, the LLM was prompted to convert the candidate triple into a natural language sentence. The sentence and candidate triple were then provided in another LLM call to determine if they had the same meaning, with the intention of nonsensical candidate tails translating poorly into a sentence and therefore not matching the original meanings.

RQ3 For question three, to understand the tradeoff between the flexibility and inconsistency of an LLM, we further conducted an ablation to compare the regex with LLM fallback we had been utilizing throughout to understand the degree to which the LLM was usefully handling what the regex could not. We similarly compared a configuration of the consensus without the LLM to equally understand if the LLM was successfully extracting and consolidating the candidate entities.

4. Experiments and Results

For our experiments we used a stratified subset of 100 samples from the LM-KBC 2025 training dataset, with examples extracted from the validation set. The sample was equivalent to about 1/5th of the entire dataset, and selected to due the long computation time of the re-generation component. The sample was randomly selected, maintaining a representative proportion for each relation type. For all LLM use, we employed Qwen3 8B through the transformers library [5] without any fine tuning or RAG. The scores we present in the following portions are the values for F1, macro precision, and macro recall. For the purpose of this investigation, we take information to mean any additional data and examples supplied alongside a prompt to the LLM. Full details of our implementation & prompts can be found in our code.

4.1. RQ1: What type of information is useful to generating candidate tails with limited access to external data?

For our first question, we began by investigating the impact of introducing LLM generated facts about the target entity. We found that the introduction of such information was not useful to the re-generation component, as the performance decreased from the base version without information, as shown in Table 1. As the primary difference between the initial generation and the re-generation was the use of examples instead of previous incorrect attempts, this is likely the reason that the initial generation improved with additional information while the re-generation did not. Therefore, the additional information may have acted primarily as noise when introduced with incorrect attempts but acted as a form of grounding with correct examples. In our next experiment, we evaluated the efficacy of positive and

Approach	F1	Precision	Recall
Base	.148	.096	.320
Intial Generation	.151	.097	.338
Re-generation	.136	.085	.340

Table 1
Addition of information describing the target entity prior to examples at generation time.

negative examples during the re-generation component. The re-generation component allows the unique opportunity to investigate whether the LLM is able to self-correct through the use of previous

[5] https://huggingface.co/Qwen/Qwen3-8B

attempts. We find that within our setting, such information does improve precision more than examples, both of which display improvement over format-only elicitation as shown in Table 2. Despite a lower precision, the correct examples yields a slightly higher F1, likely due to differing performance on certain relations, which is highlighted through the use of macro evaluation scoring. To determine the value

Approach	F1	Precision	Recall
Base	.126	.078	.325
w/ Correct Examples	.151	.097	.338
w/ Incorrect Examples (previous)	.158	.103	.338

Table 2
Re-generation experiments. The base version provides only the format without examples or additional information.

of entity/relation pair specific examples and that of relation guides, we used a baseline of relation specific examples with additional LLM generated information. We compared this base with custom examples for the entity/relation pair, and a version where a relation guide was used in place of the background information. The relation guide provided a sentence stating the expected type, for instance, for `awardWonBy` it requested a list of names, and `countryLandBordersCountry` a list of countries.

As anticipated, both the custom examples and relation guide improved on the original score as shown in Table 3. It is likely that as the examples were already specific to the relation type, there was reduced variance when introducing the specially selected examples. Had the base been relation unspecific it would likely have been possible to see a larger impact. The relation guide likely fulfilled its purpose in reducing some type irrelevant answers, which otherwise might have been difficult to achieve without additional processing.

Approach	F1	Precision	Recall
Relation Specific	.152	.101	.310
Custom for Entity/Relation	.162	.107	.335
With Relation Guide	.183	.128	.320

Table 3
Comparison of additions to initial generation. Relation specific acts as the baseline.

4.2. RQ2: Without access to a ground truth, what is an effective means for filtering poor quality triples?

In question 2 we focus on the addition of a filter to the process to remove poor quality triples. The addition of any filter showed significant improvement in precision over the base version, as displayed in Table 4. The consistent temperature judge preformed best overall in F1, over that of the judges using an average of three different temperatures, possibly due to the reduced variance of the consistent judge and non-number or nonsensical responses from the differing temperature version. Despite having good precision, the translation filter also had the lowest recall, implying that it may have been very strict in its reduction of answers, and highlighting a precision/recall tradeoff. Moreover, it involved two calls to the LLM, the first to convert the triple into a natural language sentence, and the second to compare the meaning of the two. This two step process may have introduced errors with propagated across, and decreased the number of retained candidates as a result.

Interestingly, the recall is reduced as expected across all versions with the exception of the consensus. It would be expected that in filtering out candidates the recall would decrease, however due to the use of an LLM for the consensus, it is possible that it introduced new candidates during consolidation.

Approach	F1	Precision	Recall
Base (Initial only)	.152	.101	.310
Judge	.360	.438	.305
Judge Alt. Temps	.290	.286	.295
Translate	.313	.340	.290
Consensus	.205	.147	.341

Table 4
Comparison of filtering approaches. The candidate triples were identical for each and immediately following the initial generation.

4.3. RQ3: Is the flexibility of LLM entity extraction worth the tradeoff of inconsistency?

Finally, we conduct an ablation over the initial generation to determine the efficacy of the LLM in handling the entity extraction, and a comparison of the consensus using an LLM approach and a deterministic counter approach. In order to more thoroughly process the LLM responses, regular

Approach	F1	Precision	Recall
Regex w/ LLM Fallback	.152	.101	.310
Regex only	.283	.268	.300
LLM only	.133	.088	.270

Table 5
Comparisons of the different processing of the LLM responses. We used the combination as default.

expression based extraction of numbers and capitalized sets of words, ideally proper nouns, initially was used with a fallback to an LLM if nothing was returned. We anticipated that the combination would be more effective than the regular expression alone, as the LLM might be able to extract additional valid answers. However, as the LLM only approach preformed poorly, and the version utilizing both also saw a reduction in score from the regex only version, as shown in Table 5 we hypothesize that the LLM's involvement may have contributed poor quality answers rather than simply catching differing format. Additionally, when constructing the implementation, we noticed that when the LLM generated a candidate: the phrasing, punctuation, and capitalization was often consistent each time it was generated. As such, the regex version's removal of such responses may not have actually been undesirable. When

Approach	F1	Precision	Recall
LLM Consensus	.205	.147	.341
Regex Consensus	.147	.098	.290

Table 6
Variations in consensus method.

evaluating the difference between LLM and Python counter approaches for the consensus filter, we anticipated that due to the variability of responses, the LLM approach would yield a lower score. However as can be seen in Table 6, the LLM actually yielded the highest scores. It is likely that this is due to the LLM being more flexible with it's answers and accommodating more values with the same meaning, whereas the Python counter function uses restrictive string matching.

As such, the tradeoff in between flexibility and consistency ultimately may be dependent on the setting. While cleaning answers for extraneous content may be consistent in the sorts of patterns encountered, there is a higher degree of variability in what candidate tails might be the 'same'. For instance, a regex can consistently extract a variety of proper nouns, likely better than an LLM can extract an entity without finetuning; conversely the counter function cannot handle any instances of

the same phrase with different punctuation while the LLM could recognize that both NYC and New York City are the 'same'.

5. Conclusion

Our work provides a preliminary investigation utilizing an LLM for generation, filtration, and parsing for the task of triple completion. While this work focused on the addition of information, it is important to note that variations in prompt structure could yield better use of the information, as opposed to our usage which maintained a 'promptless' approach for generating candidate tails wherever possible, with the exception of the information provided. Additionally, exploration of whether these findings are consistent across different model types and sizes could prove beneficial in future work. Our findings show, within our constrained setting, even without RAG or finetuning it is possible to improve the efficacy of LLM based AKBC. Both the introduction of additional LLM-provided information and LLM based filtration improved precision beyond simple LLM calls for completion. Additionally, we found LLM based consensus to be more effective than in-built python methods. However, we determined that a regex is more useful in handling the outputs despite the flexibility an LLM might provide.

Declaration on Generative AI

The authors have not employed any Generative AI tools in the preparation of this draft.

References

[1] J. Saad-Falcon, O. Khattab, C. Potts, M. Zaharia, ARES: An automated evaluation framework for retrieval-augmented generation systems, in: K. Duh, H. Gomez, S. Bethard (Eds.), Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 338–354. URL: https://aclanthology.org/2024.naacl-long.20/. doi:10.18653/v1/2024.naacl-long.20.

[2] S. Huo, N. Arabzadeh, C. Clarke, Retrieving supporting evidence for generative question answering, in: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region, SIGIR-AP '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 11–20. URL: https://doi.org/10.1145/3624918.3625336. doi:10.1145/3624918.3625336.

[3] J. Kalo, T. Nguyen, S. Razniewski, B. Zhang, Preface: LM-KBC challenge 2024, in: S. Razniewski, J. Kalo, S. Singhania, J. Z. Pan, T. Nguyen, B. Zhang (Eds.), Joint proceedings of the 2nd workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM 2024) and the 3rd challenge on Language Models for Knowledge Base Construction (LM-KBC 2024) co-located with the 23nd International Semantic Web Conference (ISWC 2024), Baltimore, USA, November 12, 2024, volume 3853 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3853/paper0.pdf.

[4] Y. Hu, T.-P. Nguyen, S. Ghosh, S. Razniewski, Enabling LLM knowledge analysis via extensive materialization, in: W. Che, J. Nabende, E. Shutova, M. T. Pilehvar (Eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 16189–16202. URL: https://aclanthology.org/2025.acl-long.789/.

[5] R. Cohen, M. Geva, J. Berant, A. Globerson, Crawling the internal knowledge-base of language models, in: A. Vlachos, I. Augenstein (Eds.), Findings of the Association for Computational Linguistics: EACL 2023, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 1856–1869. URL: https://aclanthology.org/2023.findings-eacl.139/. doi:10.18653/v1/2023.findings-eacl.139.

[6] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, Z. Qiu, Qwen3 technical report, 2025. URL: https://arxiv.org/abs/2505.09388. arXiv:2505.09388.

[7] L. Tang, P. Laban, G. Durrett, MiniCheck: Efficient fact-checking of LLMs on grounding documents, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 8818–8847. URL: https://aclanthology.org/2024.emnlp-main.499/. doi:10.18653/v1/2024.emnlp-main.499.

[8] R. Hu, Y. Cheng, L. Meng, J. Xia, Y. Zong, X. Shi, W. Lin, Training an llm-as-a-judge model: Pipeline, insights, and practical lessons, in: Companion Proceedings of the ACM on Web Conference 2025, WWW '25, ACM, 2025, p. 228–237. URL: http://dx.doi.org/10.1145/3701716.3715265. doi:10.1145/3701716.3715265.

[9] T. He, K. Cho, J. Glass, An empirical study on few-shot knowledge probing for pretrained language models, 2021. URL: https://arxiv.org/abs/2109.02772. arXiv:2109.02772.

[10] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, A. Miller, Language models as knowledge bases?, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), 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.

[11] Z. Jiang, F. F. Xu, J. Araki, G. Neubig, How can we know what language models know?, Transactions of the Association for Computational Linguistics 8 (2020) 423–438. URL: https://aclanthology.org/2020.tacl-1.28/. doi:10.1162/tacl_a_00324.

[12] B. Veseli, S. Singhania, S. Razniewski, G. Weikum, Evaluating language models for knowledge base completion, in: C. Pesquita, E. Jimenez-Ruiz, J. McCusker, D. Faria, M. Dragoni, A. Dimou, R. Troncy, S. Hertling (Eds.), The Semantic Web, Springer Nature Switzerland, Cham, 2023, pp. 227–243.

[13] X. Yin, B. Huang, X. Wan, ALCUNA: Large language models meet new knowledge, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 1397–1414. URL: https://aclanthology.org/2023.emnlp-main.87/. doi:10.18653/v1/2023.emnlp-main.87.

[14] L. Berglund, M. Tong, M. Kaufmann, M. Balesni, A. C. Stickland, T. Korbak, O. Evans, The reversal curse: Llms trained on "a is b" fail to learn "b is a", 2024. URL: https://arxiv.org/abs/2309.12288. arXiv:2309.12288.

[15] Y. Chen, Z. Wen, G. Fan, Z. Chen, W. Wu, D. Liu, Z. Li, B. Liu, Y. Xiao, MAPO: Boosting large language model performance with model-adaptive prompt optimization, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 3279–3304. URL: https://aclanthology.org/2023.findings-emnlp.215/. doi:10.18653/v1/2023.findings-emnlp.215.

[16] Q. Wu, M. A. Khan, S. Das, V. Nanda, B. Ghosh, C. Kolling, T. Speicher, L. Bindschaedler, K. Gummadi, E. Terzi, Towards reliable latent knowledge estimation in llms: Zero-prompt many-shot based factual knowledge extraction, in: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM '25, ACM, 2025, p. 754–763. URL: http://dx.doi.org/10.1145/3701551.3703562. doi:10.1145/3701551.3703562.

[17] J. Wiland, M. Ploner, A. Akbik, BEAR: A unified framework for evaluating relational knowledge in causal and masked language models, in: K. Duh, H. Gomez, S. Bethard (Eds.), Findings of the Association for Computational Linguistics: NAACL 2024, Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 2393–2411. URL: https://aclanthology.org/2024.findings-naacl.155/. doi:10.18653/v1/2024.findings-naacl.155.