

Agentic Knowledge Graph Entity Linking

Marius Birmele^{1,*}, Sebastian Walter^{1,*} and Hannah Bast¹

¹University of Freiburg, Georges-Köhler-Allee 51, 79110 Freiburg im Breisgau, Germany

Abstract

We propose an agentic zero-shot approach for finding entity mentions in natural-language text and linking them to canonical entities in a given knowledge graph. Our approach is based on large language models (LLMs) using tool calls to explore the knowledge graph (KG) and to annotate the text. We implement our approach in GRASP, a framework for agentic KG-based tasks, and evaluate it on five entity-linking benchmarks including KORE50 and AIDA-CoNLL using small and large, open-source and proprietary LLMs. Our system achieves an average F₁-score over all five benchmarks of 80.3% with Kimi K2.7 Code compared to 72.6% using ReFinED, a state-of-the-art non-agentic entity linker.

Keywords

Entity Linking, Large Language Model, Knowledge Graphs, Agent

1. Introduction

End-to-end knowledge graph entity linking (EL) consists of finding entity mentions in a given natural-language text, called *entity recognition* (ER), and choosing the correct matching entity out of the entity candidates from a given knowledge graph, called *entity disambiguation* (ED).

Most state-of-the-art EL systems require training or adaptation to the KG they use. This can lead to problems when training data is not available or when the KG changes. We propose a zero-shot approach using off-the-shelf reasoning LLMs equipped with KG exploration and text annotation functions. It does not require any task-specific training, only a KG that can be queried with SPARQL. Linking behavior can be steered with instructions. Several recent systems use LLMs for entity linking, usually splitting the task into stages handled by separate components. The following systems use LLMs, but only do ED: LLMAEL [1] uses LLMs to add context for ED, OneNet [2] splits ED into multiple LLM stages, and ARTER [3] uses LLMs only in hard disambiguation cases. LELA [4] does end-to-end EL but only uses reasoning LLMs for the disambiguation stage. GRASP [5], which we extend, uses LLMs for agentic KG-based question answering and was later applied to cell entity annotation (CEA) on tables [6]. ELA [7] comes closest to our approach by using LLMs for both ER and ED, but it does not do interactive KG exploration. ReFinED [8] is a frequently used non-LLM baseline. We evaluate with ELEVANT [10], a tool for benchmarking entity linkers and analyzing their errors. To our knowledge, there is no comparable zero-shot agentic system for end-to-end entity linking on natural-language text that lets the LLM agent perform the complete task while interactively exploring the knowledge graph.

Contributions

1. We present a zero-shot, end-to-end knowledge graph entity linking approach using reasoning LLMs.
2. We implement our approach as a task in GRASP, which is available on github.com/ad-freiburg/grasp. GRASP entity linking can be tried out at grasp.cs.uni-freiburg.de/entity-linking.
3. We evaluate our system. Detailed results can be viewed on grasp.cs.uni-freiburg.de/entity-linking-evaluation, GRASP configs and logs on ad-publications.cs.uni-freiburg.de/grasp/entity-linking.
4. We extend the ELEVANT entity-linking evaluation tool with a linker that sends requests to a GRASP server in order to benchmark our system. ELEVANT is available at github.com/ad-freiburg/elegant.

ISWC 2026 Companion Volume, October 25–29, 2026, Bari, Italy

*Corresponding author, equal contribution.

✉ birmelem@cs.uni-freiburg.de (M. Birmele); swalter@cs.uni-freiburg.de (S. Walter); bast@cs.uni-freiburg.de (H. Bast)

🆔 0009-0009-0314-517X (M. Birmele); 0009-0006-2613-3209 (S. Walter); 0000-0003-1213-6776 (H. Bast)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

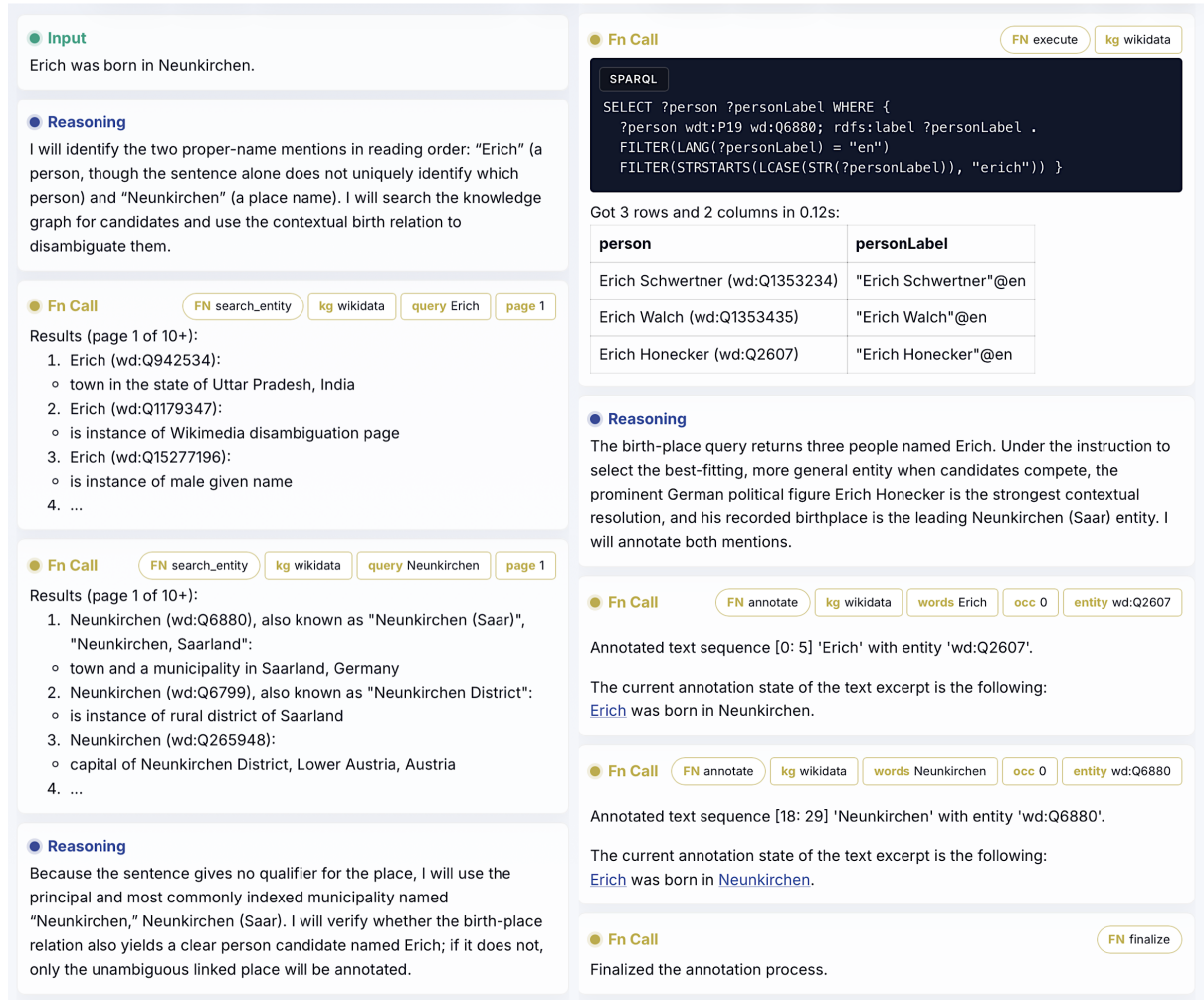


Figure 1: Example of the agentic linking process. The first reasoning step shows the LLM identifying the name “Erich” and the place “Neunkirchen”. Since the entity mentions are ambiguous, the search function call results alone are not sufficient for disambiguation. In the SPARQL query the LLM connects the found entities and then uses the annotation function to annotate them. The full example can be viewed at grasp.cs.uni-freiburg.de/share/ydsKAu.

2. Approach

We extend the functionality of GRASP with our agentic entity-linking task. GRASP is a framework that uses LLMs and tool calls for different agentic knowledge-graph-based tasks. GRASP already provides the KG exploration functions that we use. These functions are: EXE (execute SPARQL query), LST (list triples), SEN (search entity), SPR (search property), SPE (search property of entity), SOP (search object of property), SWF (search with filter).

In contrast to CEA, where a mention is a given table cell, in generic text entity linking the LLM must find the mentions itself and specify the exact linking spans rather than cells. We therefore reuse the annotation functions from CEA and change only how they address what to annotate. Since LLMs are unreliable at providing precise absolute indices of characters in a text [9], we try four annotation methods: (1) the LLM outputs the given text with the annotations in Markdown link format and we parse the output, (2) we generate indices for the words in the text and the LLM uses them to specify the words to annotate, (3) the LLM specifies the exact words to annotate and gives a prefix and suffix to disambiguate between multiple matches, and (4) the LLM specifies the words to annotate and gives their occurrence number in the excerpt. The evaluation of these methods shows that they are all within one percentage point of each other in F_1 -score, the occurrence-index method being the best. The reasoning

Table 1

Benchmarks used in the evaluation. These benchmarks are available on github.com/ad-freiburg/elevant.

Benchmark	Description	#Texts	#Entities
Wiki-Fair v2.0 (no coref)	random Wikipedia articles from 2020	120	1591
News-Fair v2.0 (no coref)	random web news articles	120	844
KORE50	sentences w/o context, hard disambiguation	50	143
MSNBC (updated)	MSNBC news articles from 2007	20	653
AIDA-CoNLL (test)	Reuters newswire articles from 1996	231	4484

traces and tool-call behavior show a specific problem for each method. The Markdown method needs the full text excerpt to be annotated in one go and requires strict adherence to the format. The indices method needs a good way to generate word indices, which is difficult and limits expressivity, since only full words can be chosen. With the prefix-suffix method the LLM often picks either too much or too little context, or reasons excessively about what the correct prefix and suffix would be. We choose the occurrence-index method for our evaluation, since it performs best and is handled most reliably by the LLM, and implement the following functions to interact with the text:

ANN: `annotate(kg: str, words: str, occ: int, entity: str)`: Finds matches of the given words to annotate in the text excerpt and annotates the match specified by the occurrence index with the entity and the knowledge graph. Example tool calls can be seen in Figure 1.

DAN: `delete(words: str, occ: int)`: Deletes the matching annotation at the given index.

SAN: `show_annotations()`: Gives the LLM the current annotation state of the text excerpt.

FIN: `finalize()`: Stops the annotation process of the text excerpt and returns the result.

The annotation process normally follows a pattern that is illustrated in the example in Figure 1. First, the LLM identifies potential entity mentions in the text. Then it uses the SEN function to search for candidates, and in cases where the search results are ambiguous or the search does not return an appropriate candidate it further refines the results by using the other KG exploration functions (mostly EXE and LST). When the LLM considers the solution satisfactory, it uses the ANN function to annotate entity mentions. Because the process is agentic, the LLM can always change the solution when it gets new insights by exploring the knowledge graph. It uses DAN to delete an annotation, SAN to show the current state, and FIN to finalize. It also uses the KG exploration functions to guide entity recognition, for example if the instructions restrict annotation to instances of certain classes.

The input to GRASP is the text to be linked, optionally restricted to an excerpt, plus special instructions. Linking excerpts of the text allows us to work in parallel, keeps the LLM conversation length shorter, and prevents the system from repeating mistakes throughout a long text. The special instructions can be useful because benchmarks use different rules for what should be linked and how entity mentions should be disambiguated.

3. Evaluation

We evaluate GRASP with seven reasoning LLMs on five entity-linking benchmarks shown in Table 1, available in ELEVANT [10], using a recent Wikidata dump from February 2026 as our KG. We compare our results with the existing results for ReFinED fine-tuned on AIDA. As an additional baseline we compare against Claude Code on KORE50, MSNBC, and Wiki-Fair-s20 (a random sample of 20 Wiki-Fair v2.0 no coref articles), prompting it to do entity linking without using GRASP while only giving it access to the same Wikidata SPARQL endpoint that we also use for our system.

For GRASP we split the benchmark texts into excerpts of 400 characters with 50 characters of overlap, providing the full text as context. The benchmark-specific instructions for KORE50, MSNBC, and AIDA-CoNLL say to link only named entities. For Wiki-Fair and News-Fair we use the linking instructions given in the paper [11] while leaving out inapplicable sections. The LLMs used for the evaluation are accessed via the OpenAI API (GPT-5.4 mini), via the Open WebUI API of the University of Freiburg

Table 2

Overall end-to-end entity-linking F_1 -score (%). Average is unweighted over all five benchmarks. LLMs are ordered by increasing average. Best value for each column in bold; second-best value underscored. Benchmark names are abbreviated. The ReFinED results are those provided by ELEVANT. The ReLiK results are taken from the paper [13].

System	Wiki-Fair	News-Fair	KORE50	MSNBC	AIDA-CoNLL	Average
ReFinED (AIDA)	68.4	66.3	68.7	74.5	<u>85.2</u>	72.6
ReLiK (AIDA)	-	-	72.8	75.0	86.4	-
GRASP Mistral Small 4	58.3	51.0	77.3	53.9	58.3	59.8
GRASP GPT-5.4 mini	73.4	69.5	80.9	74.9	72.9	74.3
GRASP Qwen3.6-27B	81.0	<u>75.9</u>	83.9	75.8	74.7	78.3
GRASP Qwen3.8-27B-INT4	80.4	74.5	84.2	<u>77.5</u>	75.9	78.5
GRASP Gemma 4 31B IT	80.9	76.4	83.3	<u>77.5</u>	76.2	78.9
GRASP Qwen3.5-397B	<u>81.4</u>	75.8	<u>85.9</u>	77.3	74.7	<u>79.0</u>
GRASP Kimi K2.7 Code	82.3	75.2	88.2	79.3	76.8	80.3

Table 3

Comparison of GRASP against Claude Code, with entity-linking F_1 -score, input tokens (T_{in}), output tokens (T_{out}), and cost. The cost for GRASP is estimated at a prefix-cache hit rate of 80% from API prices (Moonshot for Kimi K2.7 Code, OpenRouter for the self-hosted Gemma 4 31B IT). Best value for each column in bold; second-best value underscored.

System	KORE50	MSNBC	Wiki-Fair-s20	Average	T_{in}	T_{out}	Cost
Claude Code Haiku 4.5	83.1	63.6	54.6	67.1	38.3M	1.0M	<u>\$12.56</u>
Claude Code Sonnet 5	<u>86.1</u>	79.4	77.3	<u>80.9</u>	45.9M	1.3M	\$53.87
GRASP Gemma 4 31B IT	83.3	<u>77.5</u>	<u>79.2</u>	80.0	37.0M	0.7M	\$3.95
GRASP Kimi K2.7 Code	88.2	<u>79.3</u>	79.7	82.4	41.3M	0.7M	\$16.92

(Mistral Small 4, Qwen3.5-397B, Kimi K2.7 Code) and are self-hosted with vLLM [12] (Qwen3.6-27B and Gemma 4 31B IT, each on one NVIDIA H200, Qwen3.8-27B-INT4 on one NVIDIA RTX 5090). We enable reasoning mode for every LLM.

Evaluation Results

The evaluation shows that GRASP performs better than ReFinED with most LLMs on Wiki-Fair, News-Fair, KORE50, and MSNBC, as can be seen in Table 2. It performs worse on AIDA-CoNLL, which we think can be partially explained by ambiguities in linking spans, metonym and demonym disambiguation, for which ReFinED fine-tuned on AIDA is better adapted. We test this hypothesis by adding simple additional instructions on ER and linking spans (only capitalized words), metonym disambiguation (link with the implied meaning, e.g., a football team instead of the country), and demonyms (link to the country), resulting in an F_1 -score of 81.4% with Qwen3.6-27B, an improvement of 6.7 percentage points.

The results of the comparison with Claude Code can be seen in Table 3. GRASP with Kimi K2.7 Code and Claude Code with Sonnet 5 reach similar scores, but the estimated cost of GRASP is substantially lower. Claude Code with the smaller Haiku 4.5 falls behind GRASP with Gemma 4 31B IT, which suggests that a task-specific scaffold helps smaller LLMs.

To test repeatability and the effect of different KG exploration function sets, we perform ten runs with different random seeds for generation with the full function set and one run with the reduced function set with only EXE and LST. The results in Table 5 show that KG exploration without the search functions is possible, but is less efficient and leads to slightly worse results. The ten runs with the full function set have a standard deviation of 1.4 percentage points and a range of 82.2% to 86.6% F_1 -score.

Table 4 shows token usage and tool-call behavior. Both vary widely between LLMs; some explore far more than others. For a fair comparison with ReFinED, we also need to compare time and resources.

Table 4

Input tokens (T_{in}), output tokens (T_{out}), generation time (t), and function calls per LLM, summed over the five benchmarks of Table 2. OTH = {SWF, SPR, SPE, SOP}; calls to **FIN** are 1.6k for every LLM.

GRASP w/:	T_{in}	T_{out}	t	SEN	EXE	LST	OTH	ANN	DAN	SAN
Mistral Small 4	200.8M	3.4M	27.8h	11.6k	228	20	202	9.1k	84	56
GPT-5.4 mini	151.3M	1.6M	12.1h	9.2k	47	18	144	8.7k	107	34
Qwen3.6-27B	465.6M	7.8M	53.5h	13.4k	5.8k	2.7k	123	10.4k	270	821
Qwen3.8-27B-INT4	356.4M	7.8M	46.7h	10.4k	2.8k	1.8k	222	10.1k	134	643
Gemma 4 31B IT	257.8M	4.2M	42.7h	10.3k	1.1k	1.1k	33	10.5k	230	15
Qwen3.5-397B	396.0M	7.5M	38.9h	12.0k	2.0k	3.2k	180	10.6k	319	1.1k
Kimi K2.7 Code	273.5M	4.6M	30.3h	11.5k	3.5k	482	384	10.1k	92	25

Table 5

KG exploration function sets (FN), all vs. base = {EXE, LST}, and repeatability, with Qwen3.6-27B on KORE50. The values for all are averaged over ten runs with different random seeds, F_1 with its standard deviation. Calls to **FIN** are 50 for all runs.

FN	F_1	T_{in}	T_{out}	t	SEN	EXE	LST	OTH	ANN	DAN	SAN
all	83.9 ± 1.4	6.2M	130.0k	0.7h	261.8	72.3	69.2	13.2	153.0	0.4	12.4
base	80.0	12.3M	253.3k	1.5h	0	923	167	0	154	1	22

ReFinED needs 15 seconds [8] for the AIDA-CoNLL benchmark on one V100 GPU; GRASP with the INT4-quantized Qwen3.8-27B, which fits into about 18 GB of memory and thus onto a comparable card, would need about two hours at a parallelism of ten. Because we need a controlled Wikidata version, we host our own Wikidata SPARQL endpoint on a machine with a Ryzen 9 processor and 128 GB of RAM.

We find GRASP performs well in hard disambiguation tasks where preexisting world knowledge is needed or must be acquired by exploring the KG. This shows most clearly on KORE50, which consists of hard-to-disambiguate sentences that provide no contextual clues. It also works well when the instructions differ slightly from common ER instructions, like in the Fair benchmarks. Common failure cases include using incorrect linking spans, reasoning errors, and instruction-following failures in all aspects of the task, including tool calling. Borderline cases are common as well: given “David and Victoria named their children Brooklyn, Romeo, Cruz, and Harper Seven”, the model links the four children too, which our instructions allow but a benchmark’s rules may not. The full example can be viewed at grasp.cs.uni-freiburg.de/share/yambfV. Model-specific behavior is another challenge: some models give up too early, others overthink otherwise correct solutions.

4. Conclusion

We propose an agentic zero-shot LLM-based approach for end-to-end knowledge graph entity linking and implement it by extending the GRASP framework. We evaluate it with seven LLMs on five benchmarks and compare it to existing entity linkers, where it is competitive. We show that our system can also be used locally on a single GPU with good results.

For future work, we want to evaluate the system on other knowledge graphs, apply it to related tasks such as property linking, class linking, fact linking, and co-reference resolution, perform a more in-depth error analysis, and fine-tune an LLM with reinforcement learning for better instruction following.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 499552394 – SFB 1597.

Declaration on Generative AI

Our approach and the Claude Code baseline are themselves based on generative AI, i.e. large language models, so their use is inherent to this work. In addition, during the preparation of this paper the authors used the Overleaf spelling and grammar check for grammar and spelling checking, and Claude Opus 5 (Anthropic) to improve the writing style and to proofread and review the manuscript. In all cases, the authors carefully checked the AI-generated writing and take full responsibility for the content of this paper.

References

- [1] A. Xin, Y. Qi, Z. Yao, F. Zhu, K. Zeng, B. Xu, L. Hou, J. Li, LLMAEL: large language models are good context augmenters for entity linking, in: CIKM, ACM, 2025, pp. 3550–3559. doi:10.1145/3746252.3761156.
- [2] X. Liu, Y. Liu, K. Zhang, K. Wang, Q. Liu, E. Chen, OneNet: A fine-tuning free framework for few-shot entity linking via large language model prompting, in: EMNLP, Association for Computational Linguistics, 2024, pp. 13634–13651. doi:10.18653/V1/2024.EMNLP-MAIN.756.
- [3] Y. Li, A. Galimov, M. D. Ganapaneni, P. Thejaswi, D. Meng, P. Kumar, S. Potdar, Leveraging the power of large language models in entity linking via adaptive routing and targeted reasoning, in: EMNLP (Industry Track), Association for Computational Linguistics, 2025, pp. 871–882. doi:10.18653/V1/2025.EMNLP-INDUSTRY.59.
- [4] S. Haffoudhi, N. Dobricic, F. M. Suchanek, N. Holzenberger, LELA: an end-to-end llm-based entity linking framework with zero-shot domain adaptation, CoRR abs/2605.26956 (2026). doi:10.48550/ARXIV.2605.26956.
- [5] S. Walter, H. Bast, GRASP: generic reasoning and SPARQL generation across knowledge graphs, in: ISWC (1), volume 16140 of *Lecture Notes in Computer Science*, Springer, 2025, pp. 271–289. doi:10.1007/978-3-032-09527-5_15.
- [6] S. Walter, H. Bast, Knowledge graph entity linking via interactive reasoning and exploration with GRASP, in: Proceedings of the 20th International Workshop on Ontology Matching co-located with the 24th International Semantic Web Conference (ISWC 2025), Nara, Japan, November 2, 2025, CEUR-WS.org, 2025, pp. 242–248. URL: <https://ceur-ws.org/Vol-4144/om2025-SemTab-paper3.pdf>.
- [7] Y. Luo, Y. Wu, M. Li, F. Mo, J. A. Sun, X. Wang, L. Ma, Y. Zhang, J. Nie, An entity linking agent for question answering, CoRR abs/2508.03865 (2025). doi:10.48550/ARXIV.2508.03865.
- [8] T. Ayoola, S. Tyagi, J. Fisher, C. Christodoulopoulos, A. Pierleoni, ReFinED: an efficient zero-shot-capable approach to end-to-end entity linking, in: NAACL-HLT (Industry Papers), Association for Computational Linguistics, 2022, pp. 209–220. doi:10.18653/V1/2022.NAACL-INDUSTRY.24.
- [9] O. Uzan, Y. Pinter, CharBench: evaluating the role of tokenization in character-level tasks, in: Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20–27, 2026, AAAI Press, 2026, pp. 33296–33304. doi:10.1609/AAAI.V40I39.40615.
- [10] H. Bast, M. Hertel, N. Prange, ELEVANT: A fully automatic fine-grained entity linking evaluation and analysis tool, in: EMNLP (Demos), Association for Computational Linguistics, 2022, pp. 72–79. doi:10.18653/V1/2022.EMNLP-DEMOS.8.
- [11] H. Bast, M. Hertel, N. Prange, A fair and in-depth evaluation of existing end-to-end entity linking systems, in: EMNLP, Association for Computational Linguistics, 2023, pp. 6659–6672. doi:10.18653/V1/2023.EMNLP-MAIN.411.
- [12] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, I. Stoica, Efficient memory management for large language model serving with pagedattention, in: SOSP, ACM, 2023, pp. 611–626. doi:10.1145/3600006.3613165.
- [13] R. Orlando, P. H. Cabot, E. Barba, R. Navigli, ReLiK: retrieve and LinK, fast and accurate entity

linking and relation extraction on an academic budget, in: ACL (Findings), volume ACL 2024 of *Findings of ACL*, Association for Computational Linguistics, 2024, pp. 14114–14132. doi:10.18653/V1/2024.FINDINGS-ACL.839.