

TF-IDF Example Retrieval and Execution-Guided Selection for Few-Shot Text-to-SPARQL over Wikidata with Open-Weight LLMs^{*}

Emeric Cyrille Kenmegne Fokam^{1,*}, Gilbert Ebenezer Tela Tela¹

¹Department of Computer Science, University of Yaoundé I, Yaoundé, Cameroon

Abstract

We report on our participation in the two *with-mentions* tracks (English and Spanish) of the WikiKGQA @ ISWC 2026 challenge, whose task is to translate a natural-language question into a SPARQL query executable against a Wikidata subgraph. Our system chains four stages: a TF-IDF retriever that selects three similar training examples, a few-shot prompt sent to an open-weight LLM, generation of five candidates with mixed temperatures (one deterministic, four at 0.7), and selection of the first candidate returning a non-empty result on the challenge endpoint. On the development split (100 questions per language) we obtain macro F1 = 0.634 in English and 0.607 in Spanish; on the official Codabench test (75 questions per language) scores reach 0.650 and 0.630. An error analysis isolates two resistant categories: type-filter questions in “*which/qué*”, which rely on the P31/P279* chain that is incomplete on the endpoint, and COUNT queries, for which the LLM frequently forgets the aggregate. The pipeline uses only open-weight models (gpt-oss-120b, llama-3.3-70b) served free of charge through Groq Cloud.

Keywords

Knowledge Graph Question Answering, Wikidata, Large Language Models

1. Introduction

Knowledge Graph Question Answering (KGQA) aims at querying an RDF graph in natural language. On a collaborative graph such as Wikidata [1], the difficulty is twofold: mentions must first be disambiguated to the right Q-identifiers, and the logical structure of the question then translated into a coherent graph pattern. This second step involves picking the right properties and handling aggregates, filters and joins, all in a multilingual setting. The WikiKGQA @ ISWC 2026 challenge provides 492 bilingual (English/Spanish) question-SPARQL pairs built on top of QAWiki, a dedicated QLever [2] endpoint serving a simplified view of Wikidata frozen for the duration of the challenge, and an online Codabench evaluation using QALD macro F1 [3]. We participate in the two *with-mentions* tracks, where the Q-identifiers of the mentioned entities are provided with the question, isolating the query-generation problem. Our contributions are: (i) a four-stage pipeline (TF-IDF retrieval, few-shot prompting, multi-candidate generation, and execution-guided selection) built entirely on open-weight LLMs; (ii) a dev and test evaluation of both tracks, with macro F1 of 0.634/0.607 on dev and 0.650/0.630 on test; (iii) a structured error analysis that isolates the two most resistant question categories, namely type-filter questions and COUNT queries.

2. Related Work

The QALD series [4] has set the evaluation protocol still used today (macro F1 on the union of answers), and QALD-10 [3] marked its migration to Wikidata. Classical approaches based on supervised semantic

Proceedings of the Wiki-Based Knowledge Graph Question Answering Challenge (WikiKGQA), co-located with the 25th International Semantic Web Conference (ISWC 2026), Bari, Italy, October 25-29, 2026

^{*}System paper for the WikiKGQA @ ISWC 2026 challenge, en-with-mentions and es-with-mentions tracks.

^{*}Corresponding author.

✉ emeric.kenmegne@facsciences-uy1.cm (E. C. K. Fokam); gilbert.tela@facsciences-uy1.cm (G. E. T. Tela)

ORCID 0009-0001-4232-2361 (E. C. K. Fokam)



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

parsers or hand-written templates score well on distributions seen at training time but generalise poorly to new reformulations; end-to-end *text-to-SPARQL* generation with a fine-tuned seq2seq model produces syntactically valid queries but suffers from hallucinated IRIs: the model generates plausible-looking properties that do not exist in the target graph. Since GPT-3 [5], several works have shown that an instruction-tuned LLM produces competitive SPARQL queries from a handful of in-context examples [6], and KAPING [7] further injects the graph neighbourhood in the prompt. We inherit from this line the retrieval-guided few-shot principle but deliberately choose an unlearned retriever (TF-IDF [8]) so that the pipeline remains reproducible from public resources. In parallel, Wang *et al.* [9] popularised the idea that sampling multiple outputs and voting improves LLM robustness; Self-Refine [10] extends this family by asking the model to critique and rewrite its own output. Our execution-guided selection follows the same spirit but avoids the critique step: the SPARQL endpoint itself acts as an oracle, separating empty queries from exploitable ones.

3. Task and Data

For each natural-language question, the challenge asks for both an executable SPARQL query and the corresponding answer set: list of Q-identifiers for SELECT queries, a boolean for ASK, an integer for COUNT. Four tracks are offered by the organisers: two with entity mentions given (EN and ES) and two without. Our submission covers the two *with-mentions* tracks: within the challenge window we did not have time to run experiments on the two *no-mentions* tracks, and applying the same pipeline there (preceded by an entity linking stage) is left for future work. The QAWiki corpus is distributed as JSON derived from the QALD schema. Each item bundles an identifier, the question in both languages, the list of entity mentions (surface form followed by Q-identifier), the reference SPARQL query, and the list of expected answers (Fig. 1). Version 2026.2 [11] used in this paper provides 492 questions, split officially into 317 train, 100 dev and 75 test. The target graph is served by a challenge-specific QLever [2] endpoint. This detail matters: unlike the public Wikidata endpoint, the served view is *simplified* and some hierarchies (notably the transitive closure P31/P279*) are incomplete, so a syntactically perfect query may return an empty result simply because the queried property is not materialised. We exploit this in the execution-guided selection (Sect. 4). The official metric is QALD macro F1 [3]: for each question, F1 is the harmonic mean of precision and recall computed over the returned Q-identifiers, and the overall score is the mean of per-question F1s. We also report, for analysis purposes, the number of perfectly answered and completely missed questions.

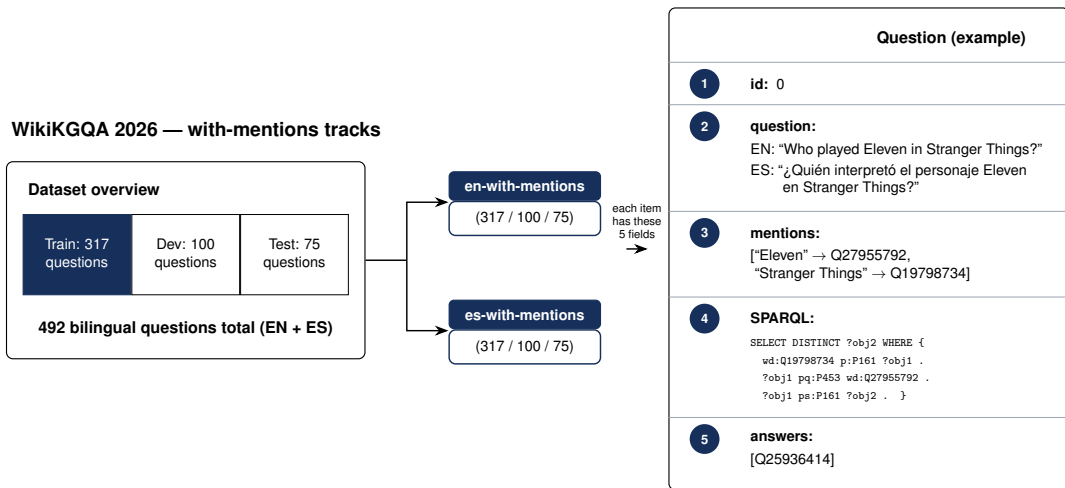


Figure 1: Format of a QAWiki item: bilingual question, entity mentions with Q-identifiers, reference SPARQL query, expected answers.

4. Proposed System

Our pipeline, referred to as P3 below, is linear and uses neither an agent nor a learned re-ranker. It takes a question and its mentions as input and produces a SPARQL query together with its answer set (Fig. 2). For each target question, the retriever selects the three nearest neighbours in the training corpus in terms of cosine similarity between TF-IDF [8] vectors built on word n -grams with $n \in \{1, 2\}$; a light heuristic infers the target query type (ASK, COUNT, SELECT) from question keywords and, at comparable similarity, favours neighbours of the same type. The vectoriser trains in under a second on the 317 training questions. The prompt consists of three blocks (Fig. 3): a system instruction fixing the expected format and forbidding PREFIX declarations (they will be prepended automatically at execution time); the three examples formatted as (question, mentions, SPARQL) triples, each wrapped in a “sparql . . . ” fenced block; the target question with its mentions in the same format. For each question we call the LLM five times: one call at zero temperature (best-guess deterministic candidate) and four calls at temperature 0.7 (diversification). Each candidate is extracted from its code block, prefixed with the standard Wikidata PREFIX declarations, and sent to the endpoint via HTTP with a 30 s timeout; 429 and 5xx codes are handled by an adaptive retry layer that respects Groq’s Retry-After. The selection rule is straightforward: we scan candidates in order (deterministic first, then stochastic) and return the answers of the first candidate producing at least one row. If all candidates return an empty set, we keep the deterministic output; this covers the rare cases where the gold answer is actually empty (negative ASK, SELECT with MINUS). No manual intervention takes place at any point: candidates are neither edited nor repaired by hand, and the selection rule is a deterministic function of the execution results alone. The pipeline does not differentiate between languages at the code level: retriever, prompt format and selection logic are identical; TF-IDF nonetheless builds two separate vocabularies, one per language.

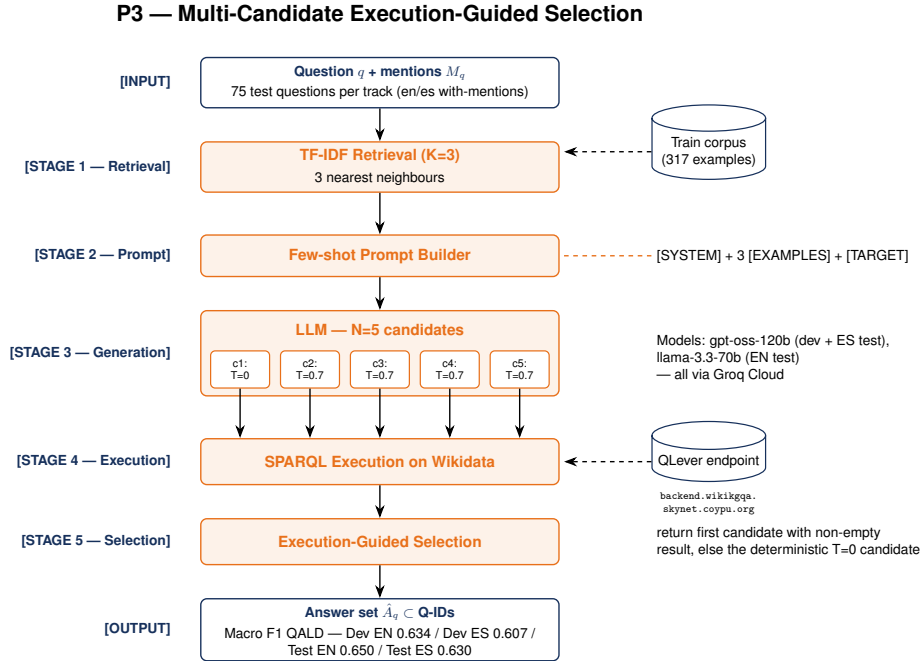


Figure 2: The P3 pipeline. (1) TF-IDF retrieval of the three nearest neighbours; (2) few-shot prompt construction; (3) LLM generation of five SPARQL candidates (one at $T=0$, four at $T=0.7$); (4) execution of every candidate on the challenge endpoint; (5) selection of the first non-empty candidate.

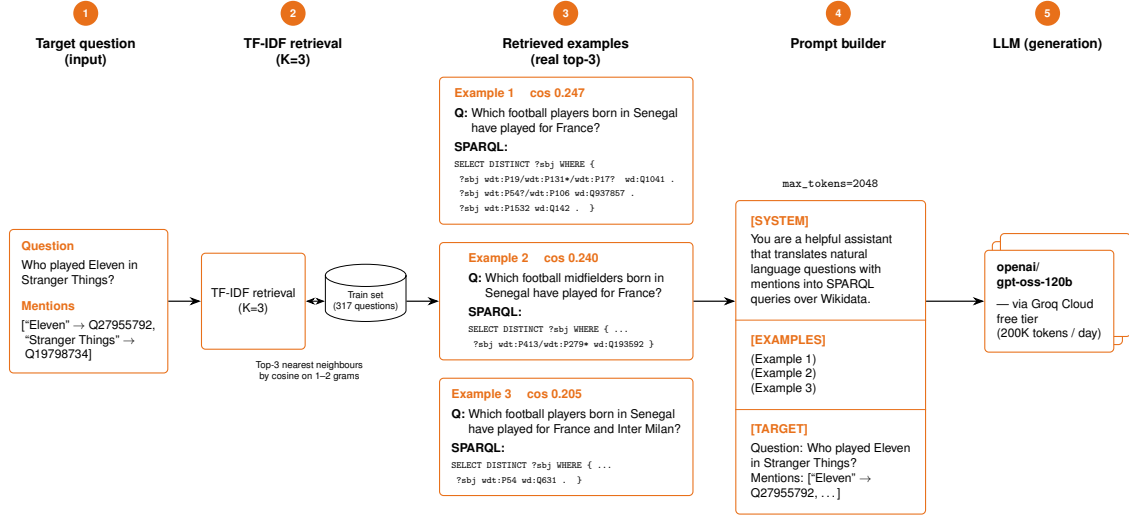


Figure 3: Few-shot prompt construction. The target question and its mentions drive TF-IDF retrieval over the 317 training examples; the three retained neighbours are concatenated with the system instruction and the target before being sent to the LLM. The three examples shown are the genuine top-3 for this question. They are topically unrelated to it (the retriever locks onto the shared *played (for)* n -grams, at cosine similarities of only 0.21–0.25) yet P3 still answers it correctly, which indicates that the retrieved examples mainly fix the output format and the Wikidata conventions rather than supply content.

5. Experimental Setup

We use the official splits (317 train, 100 dev, 75 test, for 492 questions per language in total) and evaluate with the QALD macro F1 defined above. Two open-weight LLMs power our experiments: openai/gpt-oss-120b [12] and meta-llama/llama-3.3-70b-versatile, both served via the free tier of Groq Cloud [13] with daily quotas of 200k and 100k tokens respectively. All dev scores reported in this paper (P3 and *single-shot* baseline alike) are produced by gpt-oss-120b. On the test split, the Spanish track also ran on gpt-oss-120b, but the daily TPD quota of that model had been consumed by earlier dev iterations on the day we ran the English test, so we switched to llama-3.3-70b-versatile for English to stay within a free-tier budget. Hyperparameters are fixed at $K = 3$ retrieved neighbours, $N = 5$ generated candidates, temperature sequence (0.0; 0.7; 0.7; 0.7; 0.7), a 2048-token generation budget, and a 6-second inter-call pause to respect Groq’s per-minute limit. Our baseline is the single-shot variant of P3 in which only the first, deterministic candidate is produced. We do not compare to a trained semantic parser because no trained baseline is provided by the organisers. All experiments run on a standard laptop without a GPU, with the inference load offloaded to the Groq servers. Because execution-guided selection short-circuits at the first non-empty candidate (75% of questions stop after one call, see Sect. 7), the average number of LLM calls per question drops to about two, and a full dev pass takes 15–20 minutes rather than the 50 minutes a fixed budget of five candidates would require. A pre-test verified that our local implementation of the metric reproduces exactly the score returned by the Codabench evaluator on dev submissions, allowing local iteration before every upload.

6. Results

Table 1 gathers the scores on the dev split and on the official Codabench test, and Fig. 4 offers a visual reading. The multi-candidate strategy brings a gain of +8.2 F1 points in English and only +1.6 in Spanish; the asymmetry stems from the lower bound: in Spanish the deterministic candidate alone already reaches 0.591, leaving little room for the multi-candidate rescue. Test scores (0.650 English, 0.630 Spanish) remain close to the dev scores. This parity should be read with care in English, where the test run uses a

different LLM than the dev run (Sect. 5): the two figures are therefore not a controlled measurement of generalisation. Read the other way round, the observation is arguably stronger: performance survives a change of backbone model, which suggests that it is carried by the pipeline rather than by any particular LLM.

Table 1

Macro F1 on the development split (100 questions) and the official test (75 scored questions) of both *with-mentions* tracks. The single-shot variant serves as the lower bound. All dev variants are produced by gpt-oss-120b; all scores are rounded to three decimal places.

Track	Method (LLM)	Macro F1	Perfect	Zero
EN dev	Single-shot (N=1, gpt-oss-120b)	0.552	52	40
EN dev	P3 multi-candidate (gpt-oss-120b)	0.634	60	31
ES dev	Single-shot (N=1, gpt-oss-120b)	0.591	55	37
ES dev	P3 multi-candidate (gpt-oss-120b)	0.607	56	34
EN test	P3 multi-candidate (llama-3.3-70b)	0.650	—	—
ES test	P3 multi-candidate (gpt-oss-120b)	0.630	—	—

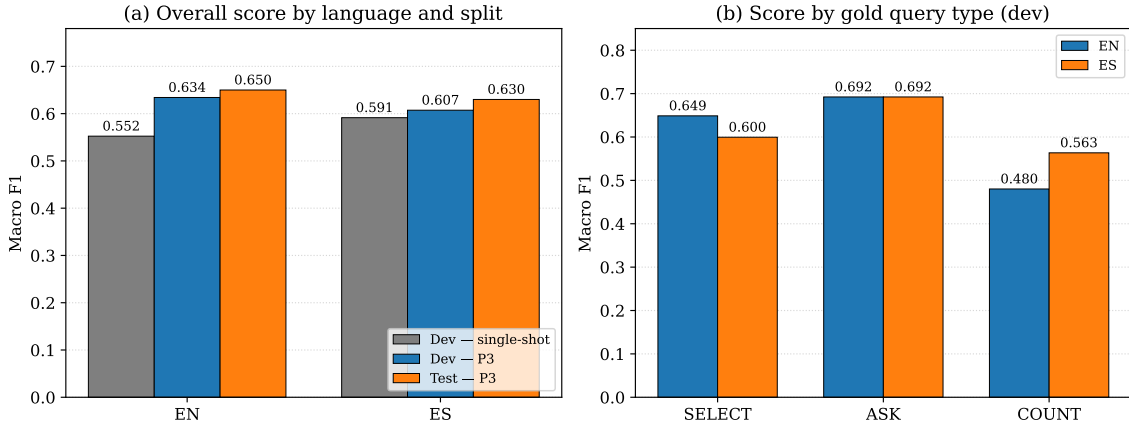


Figure 4: P3 results. (a) Macro F1 by language and split: multi-candidate is more profitable in English than in Spanish, and test is in tight parity with dev. (b) F1 by gold query type on the dev split: ASK is handled well and equally so in both languages, COUNT remains the weak spot.

To quantify how brittle the retriever is under paraphrase, we also measure the Jaccard overlap between the top-3 neighbourhoods returned by TF-IDF and those of three modern sentence encoders (all-MiniLM-L6-v2, bge-small-en-v1.5, bge-m3 [14]). Average Jaccard hovers around 0.20 in all three cases, meaning that switching to a neural retriever would massively reconstruct the few-shot context. Running the substitution end to end (regenerating every candidate with bge-m3 in place of TF-IDF and rescoring) did not fit in the challenge window, and it is the first of the directions we intend to pursue.

7. Discussions

We cross P3 predictions with the gold query type and the dominant question word (Table 2). The two views do not overlap perfectly: the top block classifies questions by the gold SPARQL type, while the bottom block classifies them by their surface form (question word). This is why the *how-many/cuántos* row does not exactly match the COUNT row: some “*which*” questions really ask for a count (“*which country has the most ... ?*”) and result in a COUNT query. COUNT queries under-perform in both languages; the recurring failure mode is dropping the aggregate: the LLM emits `SELECT ?c WHERE { . . . }` instead of `SELECT (COUNT(?c) AS ?n) WHERE { . . . }`. Conversely, ASK questions behave

identically in the two languages (0.692, 13 questions, 4 zeros on each side), which suggests that yes/no handling is governed by the pipeline rather than by the surface form of the question. The sharpest observation is structurally identical in both languages: a single interrogative category accounts for half of the zeros. In English, “*which*” questions represent 32% of the dev split and 15 zeros out of 31; in Spanish, “*qué*” questions represent 39% and 16 zeros out of 34. Both categories share the same logical structure: filter an entity set by its type, which passes through the `?x wdt:P31/wdt:P279* wd:Qtype` chain; this transitive closure is incomplete on the challenge endpoint. The LLM emits the plausible chain but execution returns zero rows. In contrast, short questions relying on a direct property (“*when/cuándo*”, “*who/quién*”) are near-perfect because they rely on well-materialised properties such as P569 (date of birth) or P161 (cast). Two complementary gradients confirm this diagnosis: F1 drops by about 0.30 point between questions with two mentions and those with three or more, and by 0.16 point between short (≤ 6 words) and medium (7–10 words) questions, both effects converging with the “*which/qué*” category which is longer and more constrained on average. Finally, the profitability of multi-candidate breaks into three regimes: about 75% of questions are solved on the first attempt; attempts 2–4 rescue effectively in English (mean F1 around 0.87) but marginally in Spanish (mean F1 around 0.30); when all five candidates return empty, no rescue happens and these 16–18 questions per language correspond to the homogeneous profile of P31/P279* chains. Three directions stand out for future work: a neural retriever (bge-m3) to capture paraphrases; a light profiling of the endpoint (one `SELECT ?p (COUNT(?o) AS ?n)` query per mentioned entity) to feed the LLM the list of actually available properties; and a single-shot self-refinement pass [10] on the specific questions where the five candidates all return empty.

Table 2

Performance by gold SPARQL query type (top block) and dominant question word (bottom block) over the 100 dev questions in EN and ES. Because both tracks share the same questions, the gold-type counts are identical across languages, which makes the EN and ES columns directly comparable. The top block sums to 100 per language; the bottom block omits an *other* bucket (5 questions in EN, 9 in ES) that groups marginal question words.

Category	EN			ES		
	<i>n</i>	F1	Zero	<i>n</i>	F1	Zero
<i>By gold SPARQL query type</i>						
SELECT	75	0.649	21	75	0.600	25
ASK	13	0.692	4	13	0.692	4
COUNT	12	0.480	6	12	0.563	5
<i>By dominant question word</i>						
which / qué	32	0.425	15	39	0.470	16
what / cuál	23	0.701	6	9	0.667	3
who / quién	13	0.822	1	13	0.800	2
when / cuándo	9	1.000	0	9	0.889	1
how-many / cuántos	5	0.200	4	10	0.400	6
yes-no	13	0.692	4	10	0.900	1

8. Conclusion

We have presented a deliberately simple pipeline for question-to-SPARQL conversion on Wikidata, applied to the two *with-mentions* tracks of the WikiKGQA @ ISWC 2026 challenge. By chaining TF-IDF retrieval, a three-example few-shot prompt, generation of five candidates at mixed temperatures, and execution-guided selection on the challenge endpoint, the system reaches macro F1 of 0.634 in English and 0.607 in Spanish on the dev split, and respectively 0.650 and 0.630 on the official Codabench test, while using only freely available open-weight LLMs. The error analysis identifies two resistant categories (type-filter questions in “*which/qué*”, defeated by the P31/P279* transitive closure being

incomplete on the endpoint, and COUNT queries for which the LLM drops the aggregate) and opens three targeted directions: a neural retriever to capture paraphrases, a light endpoint profile to avoid non-materialised properties, and a conditional self-refinement round for the questions where multi-candidate has failed to recover the answer. The code, data and Codabench archives are released at <https://github.com/emeric-cyrille/wikikgqa-iswc2026>.

Declaration on Generative AI

During the preparation of this work, the authors used **DeepL** to translate certain passages from French to English. The original version of this paper was written entirely by the authors in French. After using DeepL for translation assistance, the authors thoroughly reviewed, revised, and edited the English version to ensure accuracy, clarity, and conformity with the original intent. The authors take full responsibility for the publication's content.

References

- [1] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, *Communications of the ACM* 57 (2014).
- [2] H. Bast, B. Buchhold, Qlever: A query engine for efficient sparql+text search, in: *CIKM*, 2017.
- [3] A. Perevalov, D. Diefenbach, R. Usbeck, A. Both, Qald-10 – the 10th challenge on question answering over linked data, *Semantic Web Journal* (2022).
- [4] R. Usbeck, A.-C. Ngonga Ngomo, B. Haarmann, A. Krithara, M. Röder, G. Napolitano, 7th open challenge on question answering over linked data (qald-7), in: *SemWebEval @ ESWC*, 2017.
- [5] T. Brown, B. Mann, N. Ryder, et al., Language models are few-shot learners, in: *NeurIPS*, 2020.
- [6] T. Li, X. Ma, A. Zhuang, Y. Gu, Y. Su, W. Chen, Few-shot in-context learning on knowledge base question answering, in: *ACL*, 2023.
- [7] J. Baek, A. F. Aji, A. Saffari, Kaping: Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, in: *NLRSE Workshop @ ACL*, 2023.
- [8] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, *Information Processing & Management* 24 (1988).
- [9] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-consistency improves chain of thought reasoning in language models, in: *ICLR*, 2023.
- [10] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhume, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, P. Clark, Self-refine: Iterative refinement with self-feedback, in: *NeurIPS*, 2023.
- [11] S. Arzoo, A. Hogan, A. Moya Loustaunau, D. M. Schmidt, QAWiki v2026.2: Knowledge graph question answering (kgqa) / SPARQL query generation dataset for Wikidata, 2026. URL: <https://doi.org/10.5281/zenodo.19474009>. doi:10.5281/zenodo.19474009.
- [12] OpenAI, gpt-oss-120b: Open-weight reasoning model, 2025. Open-weight release; accessed via Groq Cloud.
- [13] Groq Inc., Groq cloud – inference platform for open-weight llms, 2024. <https://groq.com>.
- [14] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024. arXiv:2402.03216.