KNIGHT: Knowledge Graph-Driven Multiple-Choice Question Generation with Adaptive Hardness Calibration

Mohammad Amanlou[1], Erfan Shafiee Moghaddam[2], Yasaman Amou Jafari[1,*], Mahdi Noori[1,*], Farhan Farsi[3], Behnam Bahrak[4]
[1]University of Tehran [2]Independent Researcher [3]Amirkabir University of Technology [4]TEIAS Institute
Mohammad.amanlou@ut.ac.ir erfanshm12@gmail.com yasaman.jafary.a@ut.ac.ir
mahdi.noori@ut.ac.ir Farhan1379@aut.ac.ir bahrak@teias.institute
*Equal contribution.

With the rise of large language models (LLMs), they have become instrumental in applications such as Retrieval-Augmented Generation (RAG). Yet evaluating these systems remains bottlenecked by the time and cost of building specialized assessment datasets. We introduce KNIGHT, an LLM-based, knowledge-graph-driven framework for generating multiple-choice question (MCQ) datasets from external sources. KNIGHT constructs a topic-specific knowledge graph, a structured, parsimonious summary of entities and relations, that can be reused to generate instructor-controlled difficulty levels, including multi-hop questions, without repeatedly re-feeding the full source text. This KG acts as a compressed, reusable state, making question generation a cheap read over the graph. We instantiate KNIGHT on Wikipedia/Wikidata, while keeping the framework domain- and ontology-agnostic. As a case study, KNIGHT produces six MCQ datasets in History, Biology, and Mathematics. We evaluate quality on five criteria: fluency, unambiguity (single correct answer), topic relevance, option uniqueness, and answerability given the provided sources (as a proxy for hallucination). Results show that KNIGHT enables token- and cost-efficient generation from a reusable KG representation, achieves high quality across these criteria, and yields model rankings aligned with MMLU-style benchmarks, while supporting topic-specific and difficulty-controlled evaluation.

1. Introduction

Recent work identifies two main levers for LLM progress: model size and data [1]. Because scaling parameters increases financial and environmental costs (e.g., CO_2 emissions) [2, 3], attention is shifting to dataset curation; yet expert-quality datasets are expensive and slow to build, and in applied settings such as RAG and task-specific fine-tuning, public evaluation datasets remain scarce due to proprietary data despite available toolkits [4, 5]. Prior dataset-generation efforts exist [6–8], but there is still no widely adopted open-source framework that is reproducible and easy to implement. Moreover, standard MCQ benchmarks such as MMLU [9] are largely static, difficult to update, provide limited instructor-level control over per-topic difficulty, and do not expose multi-hop structure for curriculum customization. In contrast, KNIGHT enables low-cost generation of topic-specific MCQ sets with user-controlled difficulty and explicit multi-hop design, while yielding model rankings aligned with MMLU-style [10] benchmarks.

We introduce **Knowledge-graph-driven Natural Item Generation with Adaptive Hardness Tuning** (**KNIGHT**), a fully automated framework for synthesizing large-scale MCQ datasets from external document collections and ontologies with controllable difficulty. Given a user topic τ (and optional prompt), KNIGHT runs four stages: (i) *construct* a topic-specific knowledge graph (KG) via retrieval-augmented extraction [11–13], where the KG is a compact, parsimonious summary of entities and relations distilled from the sources; (ii) *generate* source-grounded MCQs by traversing multi-hop KG

Third Conference on Parsimony and Learning (CPAL 2026).

paths with configurable depth; (iii) *calibrate* difficulty based on path length and abstraction, validated via entropy-based uncertainty measures and human error patterns; and (iv) *filter* items with an LLM- and rule-based validator enforcing five criteria: grammar, single-correct-answer unambiguity, option uniqueness, answerability from evidence, and topicality [14, 15].

KNIGHT integrates RAG-based extraction, KG-guided multi-hop generation, and LLM-based validation into a reusable, modular pipeline that caches a compact topic KG for efficient dataset creation, supports forward/reverse question modes, and uses human and entropy-based checks to mitigate imperfect answerability/difficulty proxies.

We treat answerability from retrieved evidence as a proxy for generator hallucination: items judged unanswerable from the sources indicate unsupported or hallucinated content. Since the KG is built once per topic and reused across many generations, KNIGHT enables token-efficient, low-cost generation from a reusable KG representation versus naive prompting that repeatedly re-ingests long evidence contexts per question. We use **GPT-4o-mini** for all LLM calls throughout, yielding a cost- and token-aware evaluation setting.

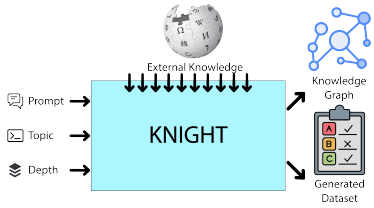

Figure 1: **KNIGHT High-level pipeline.** Given a prompt/topic and depth, KNIGHT retrieves evidence, builds a focused KG, generates MCQs, and filters them to produce the final dataset.

We instantiate the type system ϕ on the Wikipedia/Wikidata ontology (though in principle ϕ can be defined over any domain ontology or enterprise schema) and introduce KNIGHT, a token-efficient, KG-driven pipeline for generating difficulty-controlled MCQ datasets from external sources and ontologies. As case studies of its flexibility and reusability (rather than standalone benchmark contributions), we construct six Wikipedia/Wikidata-based MCQ datasets spanning history, biology, and mathematics at two difficulty levels, and run **four ablations** alongside full KNIGHT using **five GPT-4o-mini configurations** (Plain, RAG, RAG+KG, RAG+Val, and full KNIGHT). This staged comparison isolates how grounding, KG guidance, and validation affect hallucination, distractor quality, and difficulty calibration via automatic, human, and entropy-based evaluations. Items are generated within minutes on Google Colab T4 and are grammatical and difficulty-calibrated (Section 4). KNIGHT reduces hallucinations relative to Plain and RAG (Section A) and yields model rankings aligned with MMLU-style benchmarks, while being cheaper and easier to update than static MMLU-like test sets, supporting it as a scalable, low-cost benchmark generator. Our code and package are publicly available on PyPI[1] and GitHub[2].

2. Related Work

Knowledge Graph Construction. Constructing KGs from unstructured text typically uses multi-stage NLP pipelines (e.g. entity extraction/linking and relation extraction), often assuming a predefined schema and substantial supervision/training data [16, 17]. Traditional systems commonly perform named entity recognition and model-based relation extraction to identify entities and relationships, but these often require predefined schemas and extensive training. Recent LLM-based methods reduce these requirements and improve portability: Lairgi et al. [11] propose *iText2KG*, a zero-shot incremental framework with LLM-powered entity/relation extraction for topic-independent KG construction; Dessì et al. [18] extract triples from scientific abstracts via NLP/text-mining and integrate them into a KG; and Zhu et al. [19] evaluate GPT-4 [20] on KG tasks, finding that it excels at reasoning, and introduce *AutoKG*, a multi-agent LLM approach with external retrieval. Prompting has also improved relation extraction, e.g., Wikidata-informed prompts in Layegh et al. [21]. While we instantiate our type system using Wikipedia/Wikidata as an ontology [22], the mapping function ϕ can in principle be defined over other domain ontologies or schemas (e.g., enterprise KGs or specialized KBs); here we evaluate only the Wikipedia/Wikidata instantiation and leave broader generalization to future work. Finally, in line with retrieval-augmented generation, RAG combines

[1] `https://pypi.org/project/knight-mcq/`
[2] `https://github.com/ErfanShm/knight-mcq`

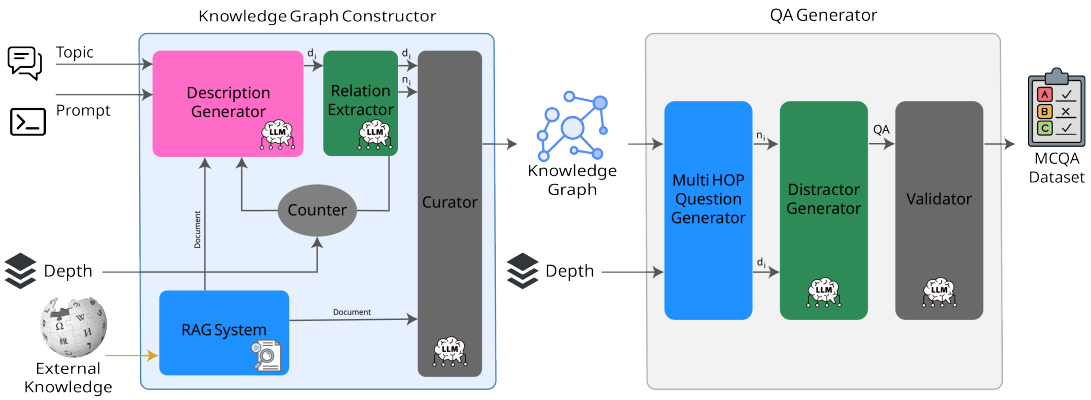

Figure 2: **KNIGHT architecture.** (Left) RAG-based KG construction: retrieve evidence, extract glosses/triples, prune to depth budget $d_{\max}$. (Right) MCQ generation: sample multi-hop paths, generate distractors, validate and filter items.

parametric LMs with retrieved knowledge bases; Lewis et al. [12] show it yields more specific, diverse, and factual outputs than parametric-only models.

Question Generation from Knowledge Graphs and Structured Data. Early work generates natural questions from KG triples using keyword extraction and RNNs [23]. Later methods go beyond single triples by encoding subgraphs with Graph2Seq and copy mechanisms [24], and by using contextual KGs with answer-aware GATs for coherent multi-hop question generation [25]. Difficulty control has also been studied explicitly: Kumar et al. [26] condition multi-hop generation on estimated KG difficulty, while Cheng et al. [27] guide reasoning complexity via step-by-step rewriting. Beyond graph-centric pipelines, *LIQUID* builds QA datasets directly from text through summarization, entity extraction, and question generation [28].

Evaluation and Filtering of Generated Questions. Ensuring the quality of generated questions requires multi-faceted evaluation, and recent work applies dedicated QA evaluation metrics. High-quality MCQs require multi-faceted evaluation: Moore et al. [29] survey metrics including LM perplexity, lexical diversity, grammar error rates, cognitive complexity, and answerability to assess fluency, uniqueness, and inferability, while Shypula et al. [30] highlight semantic diversity gains from preference-tuned LLMs. Beyond these quality dimensions, *factuality and safety* remain concerns: even strong LLMs can hallucinate and may exhibit biases, motivating automatic filtering and validation when generating educational content [31]. Factuality is further challenged by the tendency of LLMs to answer confidently even when inputs are unanswerable, motivating explicit answerability checks as a practical proxy for hallucination control [32]. In RAG, recent work also evaluates whether systems correctly *reject* unanswerable requests, complementing accuracy on answerable ones [33]. Finally, LLM-based review/validator pipelines can automatically assess MCQ validity across multiple criteria, reducing reliance on purely manual screening [34].

Building on these lines, we propose KNIGHT, a unified *end-to-end framework* that integrates KG construction, graph-driven question generation, and automatic quality filtering. A user-defined difficulty parameter controls graph depth to elicit multi-hop or higher-order items, while LLMs both generate and validate MCQs from KG paths. Compared to prior pipelines, our approach emphasizes reusable, token-efficient KG representations *and* a comprehensive LLM-powered evaluation/validation stack, aiming to produce diverse, high-quality QA pairs with improved reliability for topic-specific question sets.

3. System Design

3.1. Knowledge-Graph Constructor

Given a user-specified topic τ, optional prompt, and hardness budget $d_{\max} \in \mathbb{N}$, the Knowledge-Graph Constructor builds a directed property graph $G = (V, E, \mathcal{R})$ with canonicalized entities $v \in V$ and labeled edges $(v_h, r, v_t) \in E$, $r \in \mathcal{R}$. It iterates a retrieve–generate–filter loop (Alg. 1) combining external retrieval with LLM reasoning; the backend is swappable (HuggingFace-compatible) via a config flag.

Algorithm 1: GRAPHGENERATOR—depth-bounded KG construction

Require: seed topic/entity v_0, depth limit $d_{\max}$
$G \leftarrow$ empty graph; G.addNode(v_0)
$Q \leftarrow [(v_0, 0)]$ ▷ FIFO over (node,depth)
while $Q \neq \emptyset$ **do**
  $(v, \ell) \leftarrow$ POPFRONT(Q)
  $\mathcal{D} \leftarrow$ RETRIEVE(v)
  $\delta \leftarrow \mathcal{L}_{\text{desc}}(v, \mathcal{D})$
  $R \leftarrow \mathcal{L}_{\text{rel}}(\delta)$
  $C \leftarrow$ CURATE(R)
  for all $(v, r, u) \in C$ **do**
    G.addNode(u); G.addEdge(v, r, u)
    if $\ell + 1 \leq d_{\max}$ **then**
      PUSHBACK$(Q, (u, \ell + 1))$
    end if
  end for
end while
return G

Parsimonious, reusable representation. Once built, the KG can be cached and reused to generate many difficulty-controlled question sets (varying hop length, formats, and targets) without re-feeding long source documents, amortizing the one-time construction cost and improving token efficiency.

Evidence retrieval and description synthesis. We first retrieve a ranked context $\mathcal{D} = \{d_1, \ldots, d_k\}$ from Wikipedia (or other open sources) using dense passage retrieval and re-ranking [12, 13]. Conditioned on τ (and the optional prompt) and $\mathcal{D}$, the Description Generator $\mathcal{L}_{\text{desc}}$ produces a structured eight-point gloss δ (Appendix D).

Triple induction and deduplication. The Relation Extractor maps δ to a triple set $R(\delta)$ (Eq. 3.1) and removes near-duplicates using a Levenshtein filter with threshold $\lambda_{\max}$ [11, 18], then passes the remaining candidates to the Curator.

Curation, pruning, and depth control. The Curator applies (i) type checks (instantiated here with Wikidata), (ii) NLI-based consistency checks between node glosses and relation statements [35–37], and (iii) content-policy screening following prior safety analyses [15]. Graph expansion proceeds breadth-first and stops at depth $d_{\max}$, yielding the bounded neighborhood $V_{d_{\max}} = \{\, v \mid \text{dist}_G(v_0, v) \leq d_{\max} \,\}$ (Eq. 3.1), which matches the KG scope to the downstream MCQ generator (Section 3.2).

KG-1: Retrieval-Augmented Description Synthesis. Figure 2 shows the first stage: producing an eight-point gloss $\delta(v_0)$ for the seed v_0 via a *rank-and-generate* RAG pipeline [12, 13] with (i) a dense retriever $\mathcal{R}_{\text{enc}}$ (Contriever base; 38) encoding topic τ and scoring against a BM25-filtered corpus [39], and (ii) a cross-encoder re-ranker $\mathcal{R}_{\text{rer}}$ (MiniLM-L12; 40) refining the top–50 to k=5 passages $\mathcal{D}_0$=$\{d_1, \ldots, d_5\}$ with scores $s(d_i) \in [0, 1]$. Each retained passage d_i is injected into system-prompted **GPT-4*o*-mini**, yielding a candidate description $d_i^\star = \mathcal{L}_{\text{desc}}(\tau, d_i)$; to combine evidence we model the generation probability as a RAG mixture [12]:

$$P(d \mid \tau) = \sum_{z \in \mathcal{D}_0} P_\theta(d \mid \tau, z) \underbrace{\frac{\exp s(z)}{\sum_{z'} \exp s(z')}}_{P_{\text{ret}}(z \mid \tau)}, \tag{1}$$

where P_θ is parameterised by GPT-4*o*-mini; if $\mathcal{D}_0 = \varnothing$ (no scores >0.15), we fall back to parametric generation $P(d \mid \tau)$=$P_\theta(d \mid \tau)$. We retain a node gloss only if it is traceable to at least one retrieved passage, using:

$$\gamma(\delta) = \begin{cases} 1 & \exists z \in \mathcal{D}_0 : \texttt{overlap}(z, \delta) \geq \eta, \\ 0 & \text{otherwise}, \quad \eta = 0.35, \end{cases} \tag{2}$$

discarding $\gamma(\delta) = 0$ descriptions; this makes persistent node content externally verifiable, mitigating hallucination risk [41]. The validated description $\delta(v_0)$ is then forwarded to the Relation Extractor (§3.1), enabling breadth-first expansion up to depth $d_{\max}$ (Algorithm 1).

KG-2: Triple Induction via Relation Extraction. Given a stored gloss δ in description, we distil explicit facts using the extractor $\mathcal{L}_{\text{rel}}$, implemented with **GPT-4*o*-mini**[3], which is prompted to emit a JSON list of (*head*, *relation*, *tail*) triples. Formally, $R(\delta) = \{(h, r, t) \mid h, t \in \mathcal{E},\ r \in \mathcal{R}\}$, where $\mathcal{E}$ is the dynamic entity inventory and $\mathcal{R}$ a controlled relation schema (cf. 11, 18). A Levenshtein filter with threshold $\lambda_{\max}$ removes near-duplicate triples before insertion.

KG-3: Depth-Controlled Expansion (token-efficient). Graph growth is bounded by the hardness budget $d_{\max}$: we run breadth-first expansion with a FIFO queue over (v, ℓ) (Algorithm 1), re-applying KG-1–KG-2 while $\ell < d_{\max}$, and halt with the visited set $V_{d_{\max}} = \{v \mid \text{dist}_G(v_0, v) \leq d_{\max}\}$, ensuring no node exceeds the user-defined cognitive radius.

KG-4: Curation and pruning. Each candidate triple (h, r, t) is filtered by ϕ using (i) ontology-based type agreement (here: Wikidata) [42], (ii) NLI entailment consistency between $\delta(h)$, $\delta(t)$ and the relation phrase [35], and (iii) content-policy compliance [15]; we retain the edge iff $\phi(h, r, t) = \textsc{True}$. ϕ can be instantiated with other domain ontologies/schemas beyond Wikipedia/Wikidata.

3.2. MCQ Generator

Given a validated KG G, we generate difficulty-calibrated multi-hop MCQs in two stages: **MCQ-1** (path-conditioned synthesis) and **MCQ-2** (validation/filtering). Both use same *GPT-4$_{o\text{-}mini}$*, while the decoder is swappable via configuration.

MCQ-1: Multi-Hop MCQ Synthesis. For each seed $v_0 \in V$, we enumerate length-d forward/reverse paths (each hop in E), e.g., $P : v_0 \xrightarrow{r_1} v_1 \cdots \xrightarrow{r_d} v_d$ (or the reverse orientation). We verbalize P into a compact context template $T(P)$ by concatenating node glosses $\{\delta(v_i)\}_{i=0}^{d}$ and relation labels $\{r_i\}_{i=1}^{d}$, then prompt $\mathcal{L}_{\text{q}}$ to output an MCQ tuple $M_P = (q_P, a_P, D_P)$ with a single-sentence stem q_P, key a_P, and three semantically proximate distractors D_P [43, 44]; distractor quality is evaluated via entropy signals and human audits (§E.3).

MCQ-2: MCQ Validation & Filtering. Each candidate M_P is scored by a validator $\mathcal{L}_{\text{val}}$ on five criteria adapted from item-writing best practices [45, 46]: (i) grammatical fluency, (ii) single-key correctness, (iii) option uniqueness, (iv) answer derivability from the provided evidence (i.e., $T(P)$ and retrieved sources), and (v) topic relevance (when fixed). We retain an item iff all criteria pass, $\text{keep}(M_P) = \big[\bigwedge_{k=1}^{5} \text{criterion}_k(M_P) = \textsc{True}\big]$, discarding the rest. This LLM-as-critic loop improves factual fidelity and pedagogical validity of synthetic questions [45–47]. Retained items are serialized as JSONL with provenance metadata $\langle v_0, d, P, \text{orientation}\rangle$.

4. Experiments

4.1. Datasets

We use six domain-specific multiple-choice (MCQ) datasets as case studies to evaluate KNIGHT across three subject areas (Biology, Mathematics, History) and two difficulty levels (Level 1, Level 3): *Bio-1*, *Bio-3*, *Math-1*, *Math-3*, *Hist-1*, and *Hist-3*. The History datasets contain 241 MCQs at Level 1 and 697 at Level 3; the Biology datasets contain 323 MCQs at Level 1 and 970 at Level 3; and the Mathematics datasets contain 298 MCQs at Level 1 and 1063 at Level 3.

4.2. Experimental Setup and Baselines

All systems use the same base generator, GPT-4o-mini. For fair, evidence-grounded comparison (and to reduce hallucination), all *RAG-based* variants share the same Wikipedia retrieval step and use the retrieved passages as evidence context [12, 41, 48]. We isolate component effects by toggling retrieval grounding (RAG), topic structure (KG), and post-hoc filtering (validator), yielding five configurations: **Plain** (no evidence), **RAG** (evidence only), **RAG+KG** (evidence + topic KG; no validator), **RAG+Val** (evidence + validator; no KG), and **KNIGHT** (evidence + KG-guided multi-hop structuring + validator + difficulty control; Sec. 3). For each topic–difficulty split, we generate

[3]Checkpoint `gpt_4o_mini_2024_05`.

	Grammar Accuracy ↑					Fluency-automatic ↑					Fluency-human ↑				
Topic	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT
History	0.9994	0.9993	0.9992	0.9990	0.9989	0.9498	0.9582	0.9536	0.9549	0.9581	4.8/5	4.7/5	4.7/5	4.7/5	4.8/5
Biology	0.9992	0.9994	0.9995	0.9989	0.9998	0.9702	0.9653	0.9681	0.9591	0.9626	4.9/5	4.9/5	4.8/5	4.8/5	4.9/5
Math	0.9978	0.9986	0.9981	0.9983	0.9991	0.9711	0.9671	0.9658	0.9602	0.9685	4.9/5	4.8/5	4.7/5	4.8/5	4.7/5

Table 1: Linguistic quality aggregated over Levels. Systems: Plain GPT-4o-mini, GPT-4o-mini+RAG, GPT-4o-mini+RAG+KG, GPT-4o-mini+RAG+Validator, and KNIGHT.

N=100 MCQs per system with fixed decoding and aligned Level 1/Level 3 settings; KG-based variants additionally use the same constructed KG for comparability.

4.3. What makes a "good" MCQ dataset?

We evaluate five item-quality criteria: **linguistic quality** (well-formed, fluent text), **unambiguity** (exactly one correct key), **option uniqueness** (non-overlapping distractors), **answerability from source** (the key is derivable solely from the provided evidence), and **topic relevance** (semantic alignment with the declared topic). In the following, we evaluate these criteria across all systems (Sec. 4.2), and additionally report efficiency (generation speed) and difficulty calibration (Level 1 vs. Level 3).

4.3.1. Linguistic Quality of Questions

We assess linguistic quality to avoid surface-form confounds along three axes: *grammatical correctness*, *fluency*, and *question-length diversity*. **Grammar** is computed with LanguageTool [49, 50] as Grammar Quality$(q) = 1 - \frac{E}{W}$, where W is the number of words and E the detected errors. **Fluency** is measured both automatically with LangCheck [51], whose scores correlate with human judgments [14], and by CEFR C1/C2 annotators who rate n=100 randomly ordered items per dataset on a 5-point Likert scale (Appendix E.3). **Length diversity** is analyzed via question-length distributions [46, 47] (Appendix E.1). Table 1 reports grammar accuracy and fluency (automatic/human), aggregated over Levels, across topics and systems; overall linguistic quality is uniformly high, suggesting later differences primarily reflect grounding, structure, and validation rather than surface-form artifacts.

4.3.2. Unambiguity, Answerability, and Option Uniqueness

MCQ validity hinges on three properties: **unambiguity** (exactly one correct key), **evidence-grounded answerability** (the key is derivable from the provided evidence), and **non-overlapping distractors** (options are not near-duplicates). Violations inflate chance performance, undermine construct validity, and reduce score interpretability [44, 52, 53].

Human evaluation (Appendix E.3). For each split (topic×difficulty), blinded domain experts audited n=100 items per system and flagged four error types: REPEATED, SINGLE_KEY, OPTION_UNIQUENESS, and ANSWERABLE (key not justifiable from supplied evidence). We report counts per 100 items in Table 2 (lower is better); evidence for judgments matches system inputs (none for **Plain**; retrieved passages for **RAG**/**RAG+Val**; passages+KG context for **RAG+KG**/KNIGHT).

Answerability as a hallucination proxy. We treat ANSWERABLE violations as hallucination proxies: if the key is not derivable from the supplied evidence, the item is effectively ungrounded. Accordingly, **Plain** shows substantially higher ANSWERABLE counts, underscoring the role of retrieval grounding.

Results and component-wise patterns. Table 2 shows that KNIGHT yields the cleanest item banks overall (low repetition, fewer ambiguity errors, stronger distractor separability, and fewer unanswerable items) across both Level 1 and Level 3; importantly, Level 3 does not substantially increase violations, suggesting difficulty control without sacrificing validity. full KNIGHT (KG structuring + validation + difficulty control) achieves the strongest validity profile under both difficulty settings.

4.3.3. Topic Relevance

We assess topical alignment using two complementary signals: (i) zero-shot MNLI-style entailment [37], treating the topic as premise and the question as hypothesis; and (ii) a large LLM in few-shot

	REPEATED ↓					SINGLE_KEY ↓					OPTION_UNIQUENESS ↓					ANSWERABLE ↓				
Split	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT
History (L1)	20	19	15	2	0	10	4	6	3	2	8	5	5	5	3	26	10	8	10	6
Biology (L1)	19	17	16	3	1	16	5	6	3	1	4	3	5	5	2	19	8	7	9	4
Math (L1)	13	14	14	1	0	11	4	6	4	2	5	4	4	4	2	20	8	6	8	5
History (L3)	14	11	9	1	1	15	6	7	7	2	7	5	5	4	3	24	13	9	12	6
Biology (L3)	15	13	12	1	1	14	5	6	5	2	6	4	4	3	1	21	10	7	8	4
Math (L3)	10	10	9	0	0	13	6	5	4	3	7	5	4	2	2	28	12	9	7	6

Table 2: Human audit flags per 100 items (lower is better). All systems use GPT-4o-mini: Plain, RAG, RAG+KG, RAG+Val, and KNIGHT.

	Topic Relevance Score (Entailment) ↑					Topic Relevance (LLM) ↑					Human TOPIC Flags ↓					Off-topic Rate (LLM ∩ Entailment) ↓				
Split	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT
History (L1)	0.9432	0.9938	0.8080	0.9945	0.8214	0.8318	0.9116	0.7301	0.9218	0.7692	7	5	17	4	8	6%	4%	19.3%	3%	10.6%
Biology (L1)	0.9156	0.9885	0.7962	0.9888	0.9053	0.8611	0.9542	0.7260	0.9619	0.7979	4	3	15	3	6	3%	1%	18.0%	1%	5.5%
Math (L1)	0.9219	0.9983	0.8103	0.9989	0.8852	0.8322	0.9092	0.8211	0.9514	0.8418	8	5	14	5	7	7%	4%	14.7%	4%	7.3%
History (L3)	0.9086	0.9975	0.5765	0.9981	0.8974	0.8115	0.8884	0.5291	0.9384	0.8124	9	6	42	5	9	8%	6%	34.4%	5%	8.1%
Biology (L3)	0.9309	0.9975	0.5971	0.9977	0.8832	0.8575	0.9205	0.5430	0.9381	0.8966	8	5	39	4	3	5%	3%	22.4%	3%	3.4%
Math (L3)	0.8996	0.9981	0.6203	0.9983	0.9849	0.8866	0.9307	0.5550	0.9438	0.8909	7	4	28	3	3	3%	2%	19.0%	2%	2.2%

Table 3: Topic relevance (entailment and LLM; ↑), expert TOPIC flags (↓), and off-topic rate as the intersection of automated checks (LLM ∩ Entailment; ↓) across systems (all with GPT-4o-mini).

mode following standard NLG practice [36, 37] with prompting exemplars [54]. For topic T and question q,

$$S(q, T) = P(\text{entailment} \mid \text{premise} = T,\ \text{hypothesis} = q). \tag{3}$$

We additionally report expert TOPIC flags and an *off-topic rate* computed from the union of the two automated checks (Table 3).

Table 3 shows that KNIGHT maintains strong topical alignment across topics and difficulty levels: entailment and LLM-based relevance remain high, off-topic rates are low, and expert TOPIC flags are rare. Overall, these results indicate that the generated MCQs stay on-topic, enabling subsequent analyses to focus on validity, distractor quality, and difficulty calibration rather than topic drift.

4.4. Generation Speed

We measure end-to-end wall-clock time per topic–difficulty split on commodity hardware (Google Colab: NVIDIA Tesla T4, 12 CPU cores; Appendix H). Level 1 completes in a few minutes (History: 212s, Math: 310s, Bio: 551s), while Level 3 remains practical (History: 852s, Math: 1226s, Bio: 2449s $\approx$ 41 min). These runtimes reflect *dataset construction* (not a single exam) and enable fast, refreshable topic-specific MCQ banks compared to longer expert curation cycles for broad static benchmarks (e.g., MMLU-style suites).

KNIGHT is token- and cost-aware by design: the topic KG is built once per topic and cached, then reused to generate many variants (levels, hop lengths, and forward/reverse formats) without repeatedly re-feeding full source documents. Consequently, the *end-to-end* token usage averages $\sim$600 *total tokens per question* (prompt+completion across generation and validation stages) in our setup, whereas naive prompting and standard RAG-only baselines repeatedly inject longer evidence passages per item, inflating context length and latency. Caching amortizes the one-time KG construction cost and keeps the marginal cost of producing additional datasets low.

5. Discussion

5.1. Evaluation of Distractor Quality via Predictive Entropy

High-quality four-option MCQs require distractors that *compete* with the key without introducing ambiguity: weak distractors make items trivial, while misleading distractors can increase SINGLE_KEY and ANSWERABLE violations in human audits (Sec. 4.3.2). Following Kim et al. [55], we quantify distractor "pull" via predictive entropy over answer choices using a small fixed probe model (`LLaMA 3.2-3B-Instruct`). Given probe logits $\mathbf{z} = (z_A, z_B, z_C, z_D)$, we compute $p_i = \exp(z_i)/\sum_{j=1}^{4}\exp(z_j)$ for $i \in \{A, B, C, D\}$ [56] and entropy $H = -\sum_i p_i \log p_i$. Higher H indicates distractors receive non-trivial probability mass (stronger competition) and should coincide with lower probe accuracy when items are genuinely harder.

Split	Mean Entropy $H \uparrow$					Std. Dev. of $H \uparrow$					Probe Acc. (%) $\downarrow$				
	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT	Plain	RAG	RAG+KG	RAG+Val	KNIGHT
History (L1)	0.0	0.0106	0.0122	0.0098	0.0134	0.0	0.0058	0.0692	0.0046	0.0855	100.00	99.00	88.00	98.00	86.83
Biology (L1)	0.0	0.0038	0.0096	0.0046	0.0189	0.0	0.0007	0.0897	0.0004	0.0803	100.00	98.00	89.00	99.00	86.21
Math (L1)	0.0	0.0021	0.0256	0.0039	0.0231	0.0	0.0004	0.0991	0.0005	0.1084	100.00	100.00	84.00	98.00	84.29
History (L3)	0.0	0.0	0.0435	0.0011	0.0489	0.0	0.0	0.1846	0.0002	0.1703	100.00	99.00	69.00	99.00	66.29
Biology (L3)	0.0	0.0017	0.0191	0.0039	0.0278	0.0	0.0003	0.1156	0.0003	0.1144	100.00	99.00	71.00	99.00	66.48
Math (L3)	0.0	0.0	0.0699	0.0021	0.0826	0.0	0.0	0.2006	0.0001	0.2288	100.00	100.00	80.00	100.00	79.02

Table 4: **Distractor competition via predictive entropy.** Mean and standard deviation of predictive entropy (H) (higher $\uparrow$ = more competitive distractors) and probe accuracy (lower $\downarrow$ = harder items) using `LLaMA 3.2-3B-Instruct` as a fixed probe.

Model	History		Biology		Math		KNIGHT Avg.	MMLU	ARC	CSQA	RACE	MedMCQA	OBQA	Bench Avg.
	L1	L3	L1	L3	L1	L3								
GPT-4o[63]	92.95	86.39	95.98	87.11	95.30	86.41	90.52[1st]	79.09	86.31	70.28	67.87	57.85	67.21	71.45[1st]
Mistral Large[64]	92.19	84.16	95.18	86.84	95.07	84.10	89.59[2nd]	68.76	72.32	55.35	70.17	43.44	58.66	61.45[2nd]
Llama3-70B-Instruct[65]	92.12	85.15	94.12	86.08	95.03	84.67	89.53[3rd]	59.67	67.09	55.49	58.21	41.67	40.94	53.85[3rd]
Claude 3 Haiku[66]	91.95	81.64	95.05	83.40	93.97	81.60	87.70	57.35	63.89	55.87	57.20	40.57	42.32	52.87
Qwen1.5 (1.8B)[67]	76.76	72.45	79.88	72.89	75.84	71.43	74.88	9.99	15.84	31.13	34.91	4.70	20.37	19.49
Gemma (2B)[68]	38.59	30.73	45.51	31.27	43.62	40.45	38.36	17.52	23.93	27.40	14.32	4.57	14.26	17.00
Human (n=200)	98.60	89.20	97.40	91.60	97.40	89.40	93.92	–	–	–	–	–	–	–

Table 5: **Benchmark utility of KNIGHT.** Accuracy (%) of multiple models on KNIGHT, alongside standard MCQ benchmarks. KNIGHT Avg. averages over domains and levels; Bench Avg. averages over external suites. Superscripts denote rank by the corresponding average.

Findings (Table 4). (*1*) *Difficulty signal* (*Level 1 → Level 3*). Across domains, KNIGHT shows the expected pattern: entropy increases from Level 1 to Level 3 while probe accuracy decreases, and Std. Dev. rises at Level 3, indicating a broader, more realistic hardness spread rather than tightly clustered difficulty. The same directionality is visible for KG-guided prompting (RAG+KG), reinforcing that H tracks intended difficulty.

(*2*) *Component-wise contrast.* Plain is degenerate, with near-zero entropy and 100% probe accuracy, consistent with non-competitive distractors. Among grounded baselines, RAG and RAG+Val remain near-zero in H with near-ceiling probe accuracy across splits, suggesting retrieval alone and validator-only filtering (without KG guidance) does not reliably induce close distractors. In contrast, RAG+KG substantially increases H and lowers probe accuracy, indicating that KG-conditioned, path-/fact-driven prompting is the main driver of semantically proximate distractors. Full KNIGHT (RAG+KG+Validator) achieves the strongest overall competition profile (higher H with lower probe accuracy) while remaining well-formed under the same validity constraints used in human audits (Sec. 4.3.2).

(*3*) *Domain patterns and spread.* Math at Level 3 exhibits the highest entropy (and correspondingly lower probe accuracy), consistent with especially close distractors; History and Biology show the same qualitative Level 1→Level 3 shift. Across topics, the larger Level 3 Std. Dev. for RAG+KG and KNIGHT suggests that KG-driven prompting yields not only harder items on average but also greater within-split difficulty diversity, whereas non-KG baselines cluster near triviality.

5.2. Are KNIGHT datasets reliable, hard, and usable as benchmarks?

Table 5 reports a controlled evaluation of multiple LLMs on KNIGHT (three domains × two difficulty levels) alongside standard MCQ benchmarks (MMLU [9], ARC [57], CSQA [58], RACE [59], MedMCQA [60], OpenBookQA [61]). We follow official (or de facto) evaluation scripts and adopt the Open-LLM-Leaderboard protocol of Myrzakhan et al. [62] to mitigate MCQ selection bias and ensure cross-model comparability; we report both a *KNIGHT average* (over domains/levels) and a *benchmark average* (over external suites).

Difficulty calibration and reliability. Across models (from GPT-4o to 2B-scale baselines), Level 3 accuracy is consistently lower than Level 1 within each domain, indicating a stable, model-agnostic difficulty separation. A 200-item human study mirrors this pattern (lower L3 vs. L1 while remaining high), supporting that items are well-posed (unambiguous keys, reasonable distractors) and that Level 3 is genuinely more demanding rather than error-prone.

Convergent validity with established benchmarks. Beyond within-domain calibration, ranking under the *KNIGHT average* closely matches the *benchmark average* (GPT-4o highest, followed by Mistral Large and Llama3-70B-Instruct, with smaller models trailing), suggesting that KNIGHT captures difficulty factors predictive of general QA competence rather than overfitting to a narrow topic distribution or prompting style.

Parsimony and cost–quality trade-off. KNIGHT is token- and cost-aware: the topic KG is constructed once and reused as a compact representation to generate many variants (different levels and item patterns) without repeatedly injecting long evidence passages. This yields low marginal cost and enables frequent benchmark refresh, whereas broad static suites require substantial expert effort and longer build cycles, and naive prompting pipelines re-consume long contexts each generation pass, increasing token and runtime overhead.

KNIGHT vs. broad static suites. KNIGHT complements wide-coverage benchmarks such as MMLU: MMLU provides standardized breadth, whereas KNIGHT offers refreshable, topic-scoped evaluation with fine-grained difficulty control and explicit multi-hop structure, while preserving rank-order agreement with established suites.

5.3. Ablation Study: Component-wise Impact

We isolate the contribution of retrieval grounding, KG guidance, and validation using the staged systems in Sec. 4.2 (PLAIN, RAG, RAG+KG, RAG+VAL, and full KNIGHT). We treat ANSWERABLE violations as a hallucination proxy (unsupported content under the system-provided evidence) and use predictive entropy (Table 4) as an automatic signal of distractor competition and controlled hardness.

Retrieval grounding (Plain → RAG). Table 2 shows that retrieval substantially reduces ANSWERABLE violations and stabilizes topical alignment (Table 3). However, RAG still exhibits near-zero entropy with near-ceiling probe accuracy (Table 4), suggesting that grounding alone does not reliably yield competitive distractors or meaningful difficulty separation.

Validation without KG (RAG → RAG+Val). Adding the validator mainly improves item validity: it sharply reduces REPEATED, SINGLE_KEY, and OPTION_UNIQUENESS errors and further lowers ANSWERABLE violations (Table 2), while preserving topicality (Table 3). Yet, as in RAG, entropy remains near zero with near-ceiling probe accuracy (Table 4), indicating that validation alone does not strengthen distractor competition or enforce calibrated hardness.

KG guidance without validation (RAG → RAG+KG). Conditioning generation on KG paths/facts is the main driver of stronger distractor competition: entropy rises markedly (especially at Level 3) and probe accuracy drops (Table 4), consistent with semantically closer distractors. Without filtering, RAG+KG can also increase topical drift and validity errors (Table 3; Table 2), motivating post-hoc constraints in the full pipeline.

Combined effect (RAG+KG → KNIGHT; RAG+Val → KNIGHT). Combining KG-guided structuring with validation yields the strongest overall validity profile: KNIGHT achieves the lowest violation rates, with consistently fewer unanswerable items across Level 1 and Level 3 (Table 2); importantly, moving to Level 3 does not substantially increase violations, suggesting difficulty control without degrading validity. KNIGHT also maintains topicality (Table 3). Finally, it preserves the KG-induced difficulty signal—higher entropy, lower probe accuracy, and increased dispersion at Level 3—while avoiding the unfiltered RAG+KG error patterns (Table 4). The contrast between RAG+VAL and KNIGHT isolates the added value of KG structure *given* the same validator: KG prompting provides an interpretable, reusable scaffold that induces competitive distractors and controllable hardness, while the validator enforces item-writing constraints and further reduces hallucination as measured by answerability.

6. Conclusion

We presented KNIGHT, a knowledge-graph–driven framework for *token-efficient* and *low-cost* generation of topic-scoped four-option MCQ datasets with *controllable difficulty*. Across three domains and two difficulty settings, KNIGHT produces linguistically polished items while substantially improving validity over retrieval-only prompting: expert audits show fewer duplicates, fewer ambiguous (multi-key) questions, stronger option uniqueness, and markedly lower unanswerable items, treating ANSWERABLE violations as a proxy for hallucination. Predictive entropy further provides a practical, model-agnostic signal of distractor competitiveness and difficulty, aligning with both human and model performance.

Beyond the specific case studies built from Wikipedia/Wikidata, the main contribution is the framework itself: a reusable KG representation that can be constructed once per topic and then leveraged to generate many question variants (levels, hop patterns, formats) at low marginal cost. This makes KNIGHT complementary to broad static benchmarks: while those suites offer standardized coverage, KNIGHT enables rapid, refreshable, syllabus-aligned evaluation with explicit multi-hop structure and instructor-controlled difficulty.

Future work includes extending KNIGHT beyond single-answer MCQs, incorporating adaptive difficulty tuning via model feedback, and strengthening robustness through adversarial evaluation and explainability. We also plan to instantiate the framework on alternative ontologies and domains, and to explore cross-lingual and multimodal settings.

References

[1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[2] Imad Lakim, Ebtesam Almazrouei, Ibrahim Abualhaol, Merouane Debbah, and Julien Launay. A holistic assessment of the carbon footprint of noor, a very large Arabic language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé, editors, *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 84–94, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.8. URL https://aclanthology.org/2022.bigscience-1.8/.

[3] Yash Goel, Ayan Sengupta, and Tanmoy Chakraborty. Position: Enough of scaling llms! lets focus on downscaling. *arXiv preprint arXiv:2505.00985*, 2025.

[4] Andrei Lopatenko. Compendium of llm evaluation methods. 2024. https://github.com/alopatenko/LLMEvaluation.

[5] ExplodingGradients. Ragas: Supercharge your llm application evaluations. https://github.com/explodinggradients/ragas, 2024.

[6] Kristiyan Vachev, Momchil Hardalov, Georgi Karadzhov, Georgi Georgiev, Ivan Koychev, and Preslav Nakov. Leaf: Multiple-choice question generation. In *European Conference on Information Retrieval*, pages 321–328. Springer, 2022.

[7] Vatsal Raina and Mark Gales. Multiple-choice question generation: Towards an automated assessment framework. *arXiv preprint arXiv:2209.11830*, 2022.

[8] Giorgio Biancini, Alessio Ferrato, and Carla Limongelli. Multiple-choice question generation using large language models: Methodology and educator insights. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, pages 584–590, 2024.

[9] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. URL https://arxiv.org/abs/2009.03300.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

[11] Yassir Lairgi, Ludovic Moncla, Rémy Cazabet, Khalid Benabdeslem, and Pierre Cléau. itext2kg: Incremental knowledge graphs construction using large language models. In *International Conference on Web Information Systems Engineering*, pages 214–229. Springer, 2024.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler Kulshreshtha, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2020. URL https://arxiv.org/abs/2005.11401.

[13] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning* (*ICML 2020*), pages 3929–3938, 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

[14] Alexander Fabbri, Wojciech Kryściński, et al. Sumeval: Re-evaluating summarization evaluation. In *Proceedings of EMNLP 2021*, 2021.

[15] Rick Rejeleene, Xiaowei Xu, and John Talburt. Towards trustable language models: Investigating information quality of large language models. *arXiv preprint arXiv:2401.13086*, 2024.

[16] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62, 2023. doi: 10.1145/3618295.

[17] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, Jose Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37, 2021. doi: 10.1145/3447772.

[18] Danilo Dessì, Francesco Osborne, Diego Reforgiato Recupero, Davide Buscaldi, and Enrico Motta. Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain. *Future Generation Computer Systems*, 116: 253–264, 2021.

[19] Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58, 2024.

[20] OpenAI, Josh Achiam, Steven Adler, and .... Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[21] Amirhossein Layegh, Amir H Payberah, Ahmet Soylu, Dumitru Roman, and Mihhail Matskin. Wiki-based prompts for enhancing relation extraction using language models. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, pages 731–740, 2024.

[22] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. doi: 10.1145/2629489.

[23] Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1036/.

[24] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Toward subgraph-guided knowledge graph question generation with graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[25] Zhenping Li, Zhen Cao, Pengfei Li, Yong Zhong, and Shaobo Li. Multi-hop question generation with knowledge graph-enhanced language model. *Applied Sciences*, 13(9):5765, 2023.

[26] Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. Difficulty-controllable multi-hop question generation from knowledge graphs. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019 – 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 382–398. Springer, 2019. doi: 10.1007/978-3-030-30793-6_22. URL https://doi.org/10.1007/978-3-030-30793-6_22.

[27] Yi Cheng, Siyao Li, Bang Liu, Ruihui Zhao, Sujian Li, Chenghua Lin, and Yefeng Zheng. Guiding the growth: Difficulty-controllable question generation through step-by-step rewriting. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing* (*Volume 1: Long Papers*), pages 5968–5978, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.465. URL https://aclanthology.org/2021.acl-long.465/.

[28] Seongyun Lee, Hyunjae Kim, and Jaewoo Kang. Liquid: a framework for list question answering dataset generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13014–13024, 2023.

[29] Steven Moore, Eamon Costello, Huy A Nguyen, and John Stamper. An automatic question usability evaluation toolkit. In *International Conference on Artificial Intelligence in Education*, pages 31–46. Springer, 2024.

[30] Alexander Shypula, Shuo Li, Botong Zhang, Vishakh Padmakumar, Kayo Yin, and Osbert Bastani. Evaluating the diversity and quality of llm generated content. *arXiv preprint arXiv:2504.12522*, 2025.

[31] Rick Rejeleene, Xiaowei Xu, and John Talburt. Towards trustable language models: Investigating information quality of large language models. *arXiv preprint arXiv:2401.13086*, 2024.

[32] Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. The curious case of hallucinatory (un)answerability: Finding truths in the hidden states of over-confident large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3607–3625, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.220. URL https://aclanthology.org/2023.emnlp-main.220/.

[33] Xiangyu Peng, Prafulla Kumar Choubey, Caiming Xiong, and Chien-Sheng Wu. Unanswerability evaluation for retrieval augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), pages 8452–8472, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.415. URL https://aclanthology.org/2025.acl-long.415/.

[34] Sérgio Silva Mucciaccia, Thiago Meireles Paixão, Filipe Wall Mutz, Claudine Santos Badue, Alberto Ferreira de Souza, and Thiago Oliveira-Santos. Automatic multiple-choice question generation and evaluation systems based on LLM: A study case with university resolutions. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2246–2260, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.154/.

[35] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of ACL*, pages 4885–4901, 2020.

[36] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 632–642. Association for Computational Linguistics, 2015. URL https://aclanthology.org/D15-1075/.

[37] Adina Williams, Nikita Nangia, and Samuel R. Bowman. Broad-coverage challenge datasets for sentence understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (*NAACL-HLT*), pages 1112–1122. Association for Computational Linguistics, 2018. URL https://aclanthology.org/N18-1101/.

[38] Gautier Izacard, Lucas Hosseini, Emmanuel De Bézenac, and Vladimir Karpukhin. Unsupervised dense information retrieval with contrastive learning. In *EMNLP*, 2022.

[39] Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 2009.

[40] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *EMNLP*, 2020.

[41] Zihan Ji and et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2023.

[42] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[43] Hao Yu, Yiming Cui, and Wanxiang Che. Large language models as distractor generators for multiple-choice qa. In *Proceedings of ACL*, 2024.

[44] Thomas M. Haladyna, Steven M. Downing, and Michael C. Rodriguez. A review of multiple-choice item-writing guidelines for classroom assessment. *Applied Measurement in Education*, 15 (3):309–334, 2002.

[45] Michael Alfertshofer, Samuel Knoedler, Cosima C Hoch, Sebastian Cotofana, Adriana C Panayi, Martin Kauke-Navarro, Stefan G Tullius, Dennis P Orgill, William G Austen Jr, Bohdan Pomahac, et al. Analyzing question characteristics influencing chatgpt's performance in 3000 usmle®-style questions. *Medical Science Educator*, pages 1–11, 2024.

[46] Xin Xu, Tong Xiao, Zitong Chao, Zhenya Huang, Can Yang, and Yang Wang. Can llms solve longer math word problems better? *arXiv preprint arXiv:2405.14804*, 2024.

[47] Andrew M Bean, Karolina Korgul, Felix Krones, Robert McCraith, and Adam Mahdi. Do large language models have shared weaknesses in medical question answering? *arXiv preprint arXiv:2310.07225*, 2023.

[48] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023. URL https://arxiv.org/abs/2310.11511.

[49] LanguageTool Developers. Languagetool: Open-source grammar, style, and spell checker. https://languagetool.org/, 2025. Accessed: 2025-10-06.

[50] language-tool-python Contributors. language-tool-python: Python wrapper for languagetool. https://pypi.org/project/language-tool-python/, 2025. Accessed: 2025-10-06.

[51] Citadel AI. Langcheck: Simple, pythonic building blocks to evaluate llm applications. https://github.com/citadel-ai/langcheck, 2023. Accessed: 2025-12-13.

[52] Steven M. Downing. The effects of violating standard item-writing principles on tests and students: The consequences are serious. *Medical Education*, 39(3):291–296, 2005.

[53] Michael C. Rodriguez. Three options are optimal for multiple-choice items: A meta-analysis of 80 years of research. *Educational Measurement: Issues and Practice*, 24(2):3–13, 2005.

[54] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[55] Eunsu Kim, Juyoung Suk, Philhoon Oh, Haneul Yoo, James Thorne, and Alice Oh. Click: A benchmark dataset of cultural and linguistic intelligence in korean, 2024. URL https://arxiv.org/abs/2403.06412.

[56] Hugo Touvron and et al. Llama: Open and efficient foundation language models. In *NeurIPS*, 2023.

[57] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), 2018. URL https://arxiv.org/abs/1803.05457.

[58] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (*NAACL*), 2019. URL https://arxiv.org/abs/1811.00937.

[59] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), 2017. URL https://arxiv.org/abs/1704.04683.

[60] Abhishek Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260, 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

[61] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), 2018. URL https://arxiv.org/abs/1809.02789.

[62] Yerbolat Myrzakhan, Nelson F. Ho, Han Liu, et al. Open llm leaderboard: Heterogeneous, dynamic, and robust evaluation of llms. *arXiv preprint arXiv:2406.07545*, 2024.

[63] OpenAI. GPT-4o: System card and model overview. https://openai.com/index/gpt-4o-system-card/, 2024. Accessed 2025-10-06.

[64] Mistral AI. Mistral large. https://mistral.ai/news/mistral-large/, 2024. Accessed 2025-10-06.

[65] Meta AI. Llama 3 model card and evaluations. https://ai.meta.com/llama/, 2024. Accessed 2025-10-06.

[66] Anthropic. Claude 3 model family: Model card and system overview. https://www.anthropic.com/claude, 2024. Accessed 2025-10-06.

[67] Yuxiao Bai, Weizhe Dai, An Yang, et al. Qwen technical report: An open large language model family. *arXiv preprint arXiv:2309.16609*, 2023.

[68] Google DeepMind and Google Research. Gemma: Open models built from the research behind gemini. `https://ai.google.dev/gemma`, 2024. Accessed 2025-10-06.

[69] Thomas Petersen, Pouya Golchin, Jinwoo Im, and Felipe PJ de Barros. Electrokinetic effects on flow and ion transport in charge-patterned corrugated nanochannels. *arXiv preprint arXiv:2510.22182*, 2025.

[70] Faezeh Dehghan Tarzjani and Bhaskar Krishnamachari. Computing the saturation throughput for heterogeneous p-csma in a general wireless network. In *2025 34th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–7. IEEE, 2025.

[71] LangChain AI. langchain: Build context-aware reasoning applications. `https://github.com/langchain-ai/langchain`, 2025.

[72] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. `https://spacy.io`, 2017.

[73] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 2020.

[74] Neo4j, Inc. Neo4j developer documentation. `https://neo4j.com/docs/`. Accessed: May 20, 2025.

[75] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380. Association for Computational Linguistics, 2018.

A. Limitations

Model choice. KNIGHT is modular and can use different LLMs for different stages. In this paper, for cost and reproducibility, we use a single model (**GPT-4o-mini**) for all LLM calls; we do not perform an exhaustive, task-wise model selection study.

Data domain. We selected History, Biology, and Mathematics to represent a broad spectrum of relational diversity, ranging from narrative-heavy event chains to abstract logical structures. However, our findings may not fully generalize to domains with low relational density, such as Physics [69] or Numerical computation [70], where knowledge is often encoded in numerical constants and first-principle equations rather than explicit entity-relation triples. In such "calculation-heavy" domains, the graph-based grounding used by KNIGHT may provide less utility than in the highly interconnected domains studied here.

Residual hallucination. Grounding and validation substantially reduce unsupported content, but do not eliminate it. We operationalize hallucination via the ANSWERABLE audit flag (Sec. 4.3.2); while KNIGHT lowers this rate compared to baselines, it remains non-zero (Table 2).

Difficulty is multi-factorial. Our primary hardness control relies on KG-based signals (e.g., multi-hop structure / graph distance), which correlate well with observed difficulty, but linguistic complexity and domain prerequisites can also affect hardness.

Evaluation scope. We evaluate only the Wikipedia/Wikidata instantiation and three domains as case studies; extending the evaluation to other corpora and ontologies is left for future work.

B. Ethics Statement

KNIGHT is released to support research and educational use via transparent, reproducible, low-cost generation of topic-scoped MCQ datasets with controllable difficulty. As with any open-source content-generation tool, it may be misused (e.g., to produce misleading content); we therefore encourage responsible use consistent with research integrity and institutional guidelines.

Our study uses only publicly available sources (Wikipedia/Wikidata) and does not involve personal or sensitive data. Users applying KNIGHT to private corpora should ensure compliance with privacy and licensing requirements and avoid including personally identifiable information.

Expert annotation targeted item quality and contained no sensitive content; exemption from IRB review was determined according to institutional guidelines. Any AI assistant usage was limited to editorial and stylistic revisions and did not contribute to research design, data collection, or analysis.

C. System Usage

In this section we first outline installation, then present the two user interfaces (API & CLI), and finally detail KG construction, generation, and validation.

C.1. Installation and Configuration

The framework targets `Python` $\geq$ `3.11` and installs in a single step:

```
$pip install knight-framework
```

All transitively required libraries are *version-pinned* in `uv.lock`, notably LangChain [71], spaCy 3 [72], Transformers 4 [73], and the Neo4j [74] Python driver; this guarantees byte-identical reproduction.

External services. A Neo4j 5.x instance provides persistent KG storage, accessed via the Bolt protocol:

```
$export NEO4J_URI=bolt://localhost:7687
$export NEO4J_USER=neo4j
$export NEO4J_PASS=<pwd>
```

Memory can be used instead of Neo4j instance by passing `backend="memory"` to the constructor which results in a non-persistent KG storage.

For item synthesis we default to GPT-4o-mini. The API key of LLM must be provided:

```
$export OPENAI_API_KEY=sk-····
```

Any HuggingFace-compatible decoder (e.g., Llama-2 [56]) can be hot-swapped by setting `lm_backend="hf"`.

C.2. Unified Workflow (API & CLI)

Both interfaces expose identical functionality; we illustrate each workflow with a minimal depth-2 example that produces ten MCQs on *Biology*.

```python
from knight import KnightFramework as kframe

kf = kframe(uri="bolt://localhost:7687",
           user="neo4j", password="neo4j")

kf.build_kg(topic="Biology", depth=2)
```

```
ds = kf.generate(prompt="multiple-choice",
                 topic="Biology", depth=2, num_q=10)

report = kf.validate(ds)
ds.to_json("bio_d2.json")
```

Now for the CLI we have:

```
$knight -topic "Biology" -prompt "multiple-choice" -depth 2 -num-q 10 -output bio_d2.json
-validate
```

C.3. Advanced Settings

All components are plug-and-play: alternative KGs (e.g., Wikidata), relation whitelists, or custom prompt templates can be swapped without touching core logic, promoting reproducible ablations. Full configuration options for replication are available in our `README.md`.

In sum, KNIGHT combines structured knowledge retrieval with controllable LLM generation to deliver fact-grounded, difficulty-calibrated MCQ datasets, suitable for both educational deployment and rigorous LLM evaluation.

D. Prompts

This section documents the prompts employed in our study, with particular emphasis on few-shot prompting techniques that enable models to perform novel tasks without parameter updates [54]. Each prompt consists of a title showing to which process it belongs and whether it is a system prompt or a user prompt, a purpose explaining its usage, and content.

Structured Term Explanation System Prompt

Purpose:
Sets the LLM's persona as a scientific subject-matter expert and defines a required 8-point structure for generating comprehensive term explanations.

Content:
You are a subject-matter expert in a scientific field. Your task is to provide detailed, thorough, and academically structured explanations about terms provided by the user. Each term should be explained exhaustively using the following structure:

1. Definition and Scope – Provide a precise, scientific definition of the term. Outline its general scope, including the boundaries and extent of its meaning and use.
2. Domains of Use – Identify all relevant scientific, technical, or professional domains where this term plays a key role. Specify the fields in which this concept is critical and explain its importance in each.
3. Subfields and Disciplines – Break the term down into its major subfields, branches, or areas of study. Provide a brief but comprehensive overview of each subfield, including key principles, practices, and contributors.
4. Key Concepts and Mechanisms – Describe the most important ideas, mechanisms, or processes associated with this term in various contexts. Explain how these ideas interconnect.
5. Real-World Applications – Discuss the major practical applications of this concept in different spheres, such as industry, healthcare, environmental science, etc.
6. Case Studies and Examples – Provide specific case studies, examples, or practical demonstrations of the term in action. Show how it is applied in real-world scenarios.
7. Related and Overlapping Terms – Identify related or similar terms and concepts. Clarify how they are connected, and explain any subtle distinctions.
8. Current Research and Trends – Briefly cover the current research directions, innovations, and debates around this concept. Mention any ongoing advancements or challenges in the field.

Your explanation should be clear, well-organized, scientifically accurate, and educational. Assume that the user is unfamiliar with the term, so explain each concept thoroughly. Use precise language and cite notable

research, when possible. Dive deeply into subtopics as needed to provide a full understanding of the term's scope and implications.

Structured Term Explanation User Prompt

Purpose:
Used when an unambiguous Wikipedia summary is found. It instructs the LLM (paired with the System Prompt above) to generate the structured explanation for a specific "`[term]`", using the "`[wikipedia_summary]`" as the primary source and optionally considering the "`[parent_term]`".

Content:
Now, please apply the structured explanation approach defined in the system prompt to explain the term: "`[term]`".
Use the following Wikipedia context as the primary source for your explanation, structuring your response according to the system prompt guidelines:

— Wikipedia Context —

"`[wikipedia_summary]`"

— End Wikipedia Context —

Also consider its relationship to the parent term "`[parent_term]`".

(Note: The last line regarding "`[parent_term]`" is conditional)

Wikipedia Title Relevance Check System Prompt

Purpose:
Sets the context for the LLM, telling it to act as a relevance classifier or "domain-specific semantic filter". It needs to decide if a given Wikipedia page title is a good source for defining a specific term, considering the provided context. It explicitly asks for a "Yes" or "No" answer only.

Content:
You are performing a relevance classification task to evaluate whether a Wikipedia page title is an appropriate definition source for a given term within a specific context.
You are expected to act as a domain-specific semantic filter.
Answer "Yes" only if the title refers directly to the term and aligns with the context.
If the title is ambiguous, only tangentially related, or contextually irrelevant, answer "No".
Respond with only one word: "Yes" or "No".

Wikipedia Title Relevance Check User Prompt

Purpose:
Provides the specific data for the LLM to evaluate: the "`[term]`" needing definition, the "`[title_guess]`" (candidate Wikipedia page title), and the "`[context_hint]`" (which could be a parent term, source text, or "general knowledge"). It reiterates the request for a 'Yes' or 'No' answer.

Content:
Context: Information related to "`[context_hint]`".
Term to define: "`[term]`".
Candidate Wikipedia Page Title: "`[title_guess]`".
Evaluate relevance and respond with only 'Yes' or 'No'.

Forward MCQ Generation System Prompt

Purpose:
Instructs the LLM to act as a "structured question generation system". Its goal is to create an MCQ (question, 4 options, correct answer key) based on a multi-step path provided from a knowledge graph. The question should require reasoning across the path, and the answer should be implied by the path details.

Content:
You are a structured question generation system. Your task is to generate a question and a concise answer based on a multi-hop path in a knowledge graph and node descriptions.

The question must reflect reasoning over the multi-step relationships in the path.
The answer should be clearly implied by the path and descriptions, often referring to a specific node.

Forward MCQ Generation User Prompt

Purpose:
Provides the LLM with the specific details needed to generate the forward MCQ: examples of the task, the actual graph "`[path_representation]`", descriptions of the "`[start_node]`" and "`[end_node]`", an optional "`[topic]`" constraint, and strict formatting instructions for the output (Question, A, B, C, D, Correct Answer key).

Content:
Follow the instructions in the system prompt to generate a multiple-choice question based on the provided path and node descriptions.

— Few-Shot —

"`[few-shot example]`"

— End Few-Shot —

IMPORTANT: The generated Question and Options MUST be relevant to the overall topic: "`[topic]`".

Now, generate for the following:
`Path: "[path_representation]"`
`Start Node: "[start_node]"`
`Description: "[start_desc]"`
`End Node: "[end_node]"`
`Description: "[end_desc]"`

IMPORTANT: You MUST generate exactly four options (A, B, C, D) and indicate the single correct answer key. Adhere strictly to the output format below.

Output:
Question: [Your generated question reflecting the multi-step path]
A) [Option A]
B) [Option B]
C) [Option C]
D) [Option D]
Correct Answer: [A, B, C, or D]

Reverse MCQ Generation System Prompt

Purpose:
Sets the LLM's role as a "reasoning assistant" focused on generating *reverse* questions. The goal is to create an MCQ where the "`[start_node]`" of the provided graph path is the correct answer. It suggests using the end node's perspective to guide the reasoning.

Content:
You are a reasoning assistant generating reverse questions from knowledge graph paths.
Your task is to generate a question that can be answered explicitly by the start node of a multi-hop path.
Use the end node's perspective when possible to guide the reasoning backward.

Reverse MCQ Generation User Prompt

Purpose:
Provides the LLM with specific instructions and data to generate the reverse MCQ. It includes examples, the graph, descriptions of "`[start_node]`" and "`[end_node]`", an optional "`[topic]`" constraint, and strict formatting instructions. Crucially, it emphasizes that the correct answer must be the "`[start_node]`".

Content:
Follow the instructions in the system prompt to generate a multiple-choice question where the start node ("`[start_node]`") is the correct answer.

— Few-Shot —

"[few-shot example]"

– End Few-Shot –

IMPORTANT: The generated Question and Options MUST be relevant to the overall topic: "[topic]".

Now, generate for the following:
Path: "[path_representation]"
Start Node: "[start_node]"
Description: "[start_desc]"
End Node: "[end_node]"
Description: "[end_desc]"

IMPORTANT: You MUST generate exactly four options (A, B, C, D) and indicate the single correct answer key (which MUST correspond to the option containing the Start Node name "[start_node]"). Adhere strictly to the output format below.

Output:
Question: [Generated question targeting the start node]
A) [Option A]
B) [Option B]
C) [Option C]
D) [Option D]
Correct Answer: [Letter corresponding to the option containing the exact text "[start_node]"]

GPT Triplet Extraction System Prompt

Purpose:
This prompt instructs the LLM to extract significant subject-predicate-object triplets from the provided text. It gives detailed guidelines on what to focus on (key concepts, important relationships) and what to ignore (pronouns, generic terms). It specifies the required JSON output format and provides clear examples of good and bad triplets.

Content:
You are an information-extraction specialist.
Extract only the most significant and meaningful "[subject-predicate-object]" triplets from any text you receive.

Here are the guidelines you should follow :

- Focus on important entities: names, places, concepts, achievements.
- Include defining characteristics and significant relationships.
- Capture major influences, contributions, and key life events.
- Skip generic pronouns, articles, and common words.
- Write relations in clear lowercase and with underscores.

IMPORTANT: The generated output must accommodate with this format.

```
{
  "triplets": [
    {
      "head": "specific_entity",
      "relation": "significant_relation",
      "tail": "important_concept"
    },
    {
      "head": "major_figure",
      "relation": "notable_achievement",
      "tail": "specific_contribution"
    }
  ]
}
```

Bellow are some of the good and bad examples:

– Few-Shot –

"[few-shot example]"

– End Few-Shot –

GPT Triplet Extraction User Prompt

Purpose:
Provides the LLM with the specific text to extract triplets based on the instructions given in system prompt.

Content:
Follow the instructions in the system prompt to extract subject-predicate-object triplets from the text below.

– Start of the text input –

"[text-content]"

– End of the text input –

MCQ Validation System Prompt

Purpose:
Defines the LLM's role as an evaluator for MCQs generated from knowledge graph paths. It needs to assess grammar/clarity, whether the correct answer key is supported *only* by the provided path details, and optionally, relevance to a given topic. It demands a specific output format.

Content:
You are MCQ-validation assistant. Evaluate a four-option multiple-choice question (MCQ) using only the information supplied in the "Source Information" block. Answer with five "[YES/NO]" (or N/A) tags in the exact order and casing shown below.

Checklist

1. GRAMMAR_FLUENCY
Is the Question spelled and phrased correctly and clearly?
2. SINGLE_CORRECT_KEY
Is exactly one option marked as correct?
3. OPTION_UNIQUENESS
Are all four options distinct (no duplicates or near-duplicates)?
4. ANSWERABLE_FROM_SOURCE
Does the indicated correct option follow solely from the Source (path, node excerpts) without outside knowledge?
5. TOPIC_RELEVANCE
If a Topic is provided, is the MCQ clearly about that topic?

MCQ Validation User Prompt

Purpose:
Provides the LLM with the specific MCQ data ("[question]", "[correct_answer_key]") and its source-details (including the "[path_representation]", "[start_node]", "[end_node]" and etc) to evaluate. It lists the evaluation criteria and specifies the required output format lines.

Content: Follow the instructions in the system prompt to evaluate the following MCQ based *only* on the Source Information.

– Few-Shot –

"[few-shot example]"

– End Few-Shot –

Now, evaluate the following question:
Question: "[question_text]"
A) [Option A]
B) [Option B]
C) [Option C]
D) [Option D]
Correct Answer: "[correct_answer_key]"

Topic (optional): "[topic_or_blank]"

Source Information
Path: "[path_representation]"
Start Node: "[start_node]"
Description: "[start_desc]"
End Node "[end_node]"
Description: "[end_desc]"

IMPORTANT: You MUST generate exactly 5 responses for each criterion based on the provided output below.

Output:
Grammar_Fluency: "[YES/NO]"
Single_Correct_Key: "[YES/NO]"
Option_Uniqueness: "[YES/NO]"
Answerable_From_Source: "[YES/NO]"
Topic_Relevant: "[YES/NO or N/A]"

Term Extraction System Prompt (Baseline pipeline)

Purpose: Defines the LLM's role as an expert at identifying key encyclopedic terms from a text and specifies strict JSON output requirements.
Content: You are an expert at identifying key encyclopedic terms from a text. Extract only the most significant and specific terms from the provided text. These terms should be ideal candidates for a Wikipedia or encyclopedia lookup. Return your answer strictly in the JSON schema shown below.
GUIDELINES 1. Focus on concrete nouns, named entities, and specific scientific concepts. 2. Keep the terms concise and specific. 3. Extract the base form of a term (e.g., cell" instead of cells"). 4. Ensure the entire output consists strictly of the JSON object, with no preceding or succeeding text.
OUTPUT FORMAT (MANDATORY)

```
"terms": [
"term1",
"term2",
"term3"
]
```

EXAMPLES (GOOD)
✓ mitochondria"
✓ Gregor Mendel"
✓ photosynthesis"
✓ natural selection"

AVOID (BAD)
× various aspects"
× complex functions"
× scientific study of life"
× living organisms"

E. Additional Evaluation Metrics

The main paper focuses on grammatical fluency, presence of a single correct answer option, answerability, and topic-relevance. Here we document *additional* metrics that were computed for every dataset but omitted from the core discussion for space and interpretability reasons.

E.1. Question Length Diversity

Recent studies indicate that question length can significantly influence LLM accuracy. Bean et al. [47] found that longer medical exam questions were associated with lower model accuracy. Similarly, Alfertshofer et al. [45] reported that ChatGPT was more likely to answer longer USMLE-style questions incorrectly. Xu et al. [46] likewise observed that LLMs achieve significantly higher accuracy in shorter math word problems. Motivated by these findings, we ensured that our generated questions

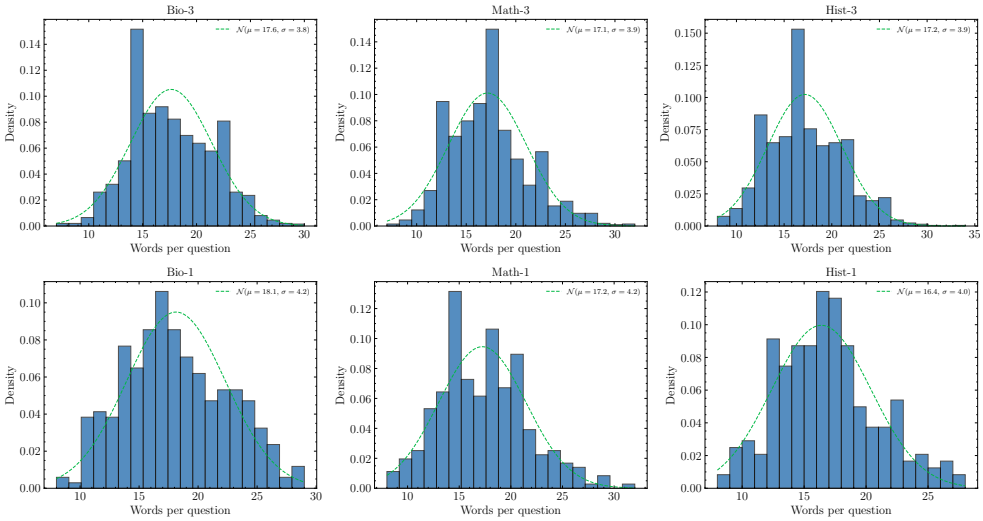

Figure 3: Distribution of question lengths for each dataset (histograms), demonstrating an approximately normal shape.

span a broad range of lengths. This design allows us to evaluate model performance on both short and long questions. Figure 3 shows the resulting distribution of question lengths for each dataset; notably, these distributions closely approximate a normal shape, indicating a balanced mix of short and long questions.

E.2. Formalizing High-Quality Four-Option MCQs

A key contribution of this work is the precise specification and validation of five core criteria that distinguish a high-quality multiple-choice question (MCQ) with exactly four options. Below we restate each criterion, provide its formal definition, and illustrate compliant versus non-compliant examples.

1. Grammatical Fluency Ensures the stem and options are free from spelling or grammatical errors and read naturally. Formally, a question q satisfies this criterion if it passes both automated grammar checks and human inspection for clarity and style.

We quantitatively assess grammatical accuracy of each question q comprising W words by detecting the number of grammatical errors E using the LanguageTool[4] system [49, 50]. The *Grammar Quality Score* is defined as

$$\text{GrammarQuality}(q) = 1 - \frac{E}{W} \tag{4}$$

which penalizes questions proportionally to the error frequency relative to length. For fluency evaluation, we employ the LangCheck toolkit [51], which estimates naturalness via normalized log-probabilities from a pretrained language model. Higher fluency scores correspond to more coherent and natural text, a correlation empirically supported by Fabbri et al. [14].

Example of Non-compliance:
`Which of the following is the` **`capital`** `city of France?`
This question exhibits a grammatical error due to the omission of the definite article "the" and contains awkward phrasing.

[4]GitHub for LanguageTool

Compliant form:
`Which of the following is the capital city of France?`
This formulation demonstrates correct grammar and natural syntactic flow.

2. Single Correct Key Exactly one option $o_k \in \{o_1, o_2, o_3, o_4\}$ is correct:

$$\exists! \, o_k : \ \text{Correct}(o_k) = \text{True}. \tag{5}$$

This avoids ambiguity in scoring and interpretation.

Example of Non-compliance: `Which are prime numbers?`
`Options:` $\{2, 3, 4, 5\}$ `(with two correct answers: 2 and 3).`
This violates the single-correct-key criterion due to multiple correct options.

Compliant form: `Which number is the smallest prime?`
`Options:` $\{2, 4, 6, 8\}$ `(only one correct answer: 2).`
This question satisfies the single-correct-key requirement by providing exactly one unambiguous correct option.

3. Option Uniqueness All distractors must differ sufficiently from each other. For options o_i, o_j:

$$\text{sim}(o_i, o_j) < \delta, \tag{6}$$

where sim is a lexical/semantic similarity metric and δ a low threshold.

Example of Non-compliance:
`Options: {“New York City”, “NYC”, “Los Angeles”, . . . }.`
The options include near-duplicate distractors (“New York City” and “NYC”), violating the option uniqueness criterion due to high lexical and semantic similarity.

Compliant form:
`Options: {“New York City”, “Los Angeles”, “Chicago”, “Houston”}.`
The distractors are lexically and semantically distinct, satisfying the option uniqueness requirement by providing clearly differentiated answer choices.

4. Answerability from Source The correct answer must be derivable solely from the provided external knowledge G and question q:

$$P(o_k \mid G, q) \ \gg \ P(o_i \mid G, q), \quad \forall i \neq k. \tag{7}$$

Example of Non-compliance:
`If` G `lacks “Eiffel Tower” data, asking “Where is the Eiffel Tower?” is invalid.`

Compliant form:
`If` G `contains “Paris is the capital of France,” asking “What is the capital of France?” is valid.`

5. Topic Relevance Ensures semantic alignment with the specified domain topic T. We compute an entailment score:

$$\begin{aligned} S(q, T) = P(&\text{entailment} \mid \\ &\text{premise} = T, \ \text{hypothesis} = q) \end{aligned} \tag{8}$$

Here a high entailment score indicates that the generated questions are strongly aligned with and highly pertinent to the specified topic.

Example of Non-compliance:
`A photosynthesis question in “World History.”`

Compliant form:
`“What sparked the outbreak of World War I?” in “World History.”`

E.3. Quantitative Analysis of Expert Annotations and Quality Flags

Dataset	GRAMMAR	SINGLE_KEY	OPTION_UNIQUENESS	ANSWERABLE	TOPIC
Hist-1	1	2	3	6	9
Bio-1	0	1	2	4	7
Math-1	2	2	2	5	8
Hist-3	2	2	3	6	10
Bio-3	1	2	1	4	4
Math-3	2	3	2	6	4

Table 6: Aggregated expert-raised flags indicating potential quality violations by dataset and criterion. Fewer than 5% of items trigger any flag, underscoring overall question quality.

Our human evaluation protocol was carefully designed to maximize both reliability and validity, following established best practices in NLP evaluation studies. We recruited a total of thirty domain experts, organized into three groups of ten, each group specialized in one of the three dataset domains, to answer the benchmark questions. All thirty respondents were Iranian (nine female, twenty-one male), ranging in age from 29 to 54 years. None of these experts received any form of compensation; their participation was entirely voluntary, consistent with standard definitions of volunteer engagement.

Each expert answered 40 questions from the dataset assigned to them. Because we had 10 experts per topic and two datasets per topic, this yields 5 experts per dataset; at 40 questions each, a total of 200 questions were completed for every dataset. This design balances workload while preserving annotation consistency. Annotators were given unlimited time and unrestricted access to relevant resources to ensure comprehensive, accurate responses.

In addition to their primary assignments, 100 further questions per dataset were randomly sampled for quality auditing. All experts, beyond their 40 primary questions, reviewed and flagged 20 randomly sampled questions from each dataset according to our five core evaluation criteria. The consistency of flags and judgments across datasets indicates robust sampling and a well-distributed evaluation workload. Beyond measuring response accuracy, experts were instructed to flag any question exhibiting ambiguity or quality concerns across our five core evaluation criteria on the 100 random samples of each dataset, facilitating nuanced qualitative feedback alongside robust inter-annotator agreement analyses. Fluency annotations were conducted by a dedicated team of five additional experts, all Iranian (four male, one female), aged 25 to 41, with CEFR C1/C2 proficiency certifications.

Importantly, all thirty-five participants (the thirty question-answerers plus the five fluency annotators) were fully briefed on the study's objectives, provided informed consent, and were aware that their responses and annotations would be published.

We computed the Pearson correlation coefficient r between human error rates E_{human} and model entropy scores H across datasets and difficulty levels, obtaining:

$$r = \frac{\text{cov}(E_{\text{human}}, H)}{\sigma_E \sigma_H} \approx 0.78, \tag{9}$$

indicating a strong positive correlation between human-perceived difficulty and model uncertainty.

Third, inter-annotator agreement, measured by Fleiss' Kappa κ, consistently exceeded 0.82 in all domains:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} > 0.82, \tag{10}$$

where $\bar{P}$ and $\bar{P}_e$ denote observed and chance agreement, respectively. This confirms high annotation reliability.

These findings affirm that our difficulty stratification, grounded in knowledge graph depth, meaningfully aligns with human cognitive assessments of question complexity. Moreover, the greater

increase in model entropy from level 1 to 3 relative to the rise in human error rates suggests that large language models possess heightened sensitivity to subtle complexity variations, pointing toward promising directions for interpretability research.

In addition to these quantitative measures, detailed quality control was conducted through expert-flagged quality violations across five criteria: grammar, single correct key, option uniqueness, answerability from source, and topic relevance. Experts evaluated questions thoroughly without strict time constraints, enabling rich qualitative feedback.

Table 6 presents the aggregated counts of expert-raised flags per criterion and dataset.

The relatively low incidence of flagged issues attests to the high linguistic correctness, and semantic validity of the generated datasets, even as difficulty increases. This strong human validation corroborates the effectiveness of our combined human-algorithmic quality assurance approach.

Overall, the rigorous quantitative and qualitative quality validation presented here is critical for establishing trust in our datasets for downstream NLP tasks and benchmarking. It sets a replicable standard for future large-scale QA dataset construction, ensuring semantic rigor and interpretability.

E.4. Significance Analyses for Topic Relevance

Goal and Setting. We compare per-item topicality between our system (KNIGHT) and a GPT4o-MINI baseline across subjects (History, Biology, Math) and difficulty levels (L1/L3). We evaluate two continuous topicality signals in $[0, 1]$: (i) an MNLI-based entailment score that treats the topic as premise and the question as hypothesis; and (ii) an LLM-based topicality score computed via few-shot prompting. Higher is better in both. The baseline set includes about 100 items per split, and KNIGHT about $1{,}000$.

Pre-processing and quality control. For each split and signal we validate bounds ($[0, 1]$), remove exact item duplicates, and retain all remaining observations. We analyze each split separately; an "overall" roll-up is provided only for descriptive context and not as a substitute for per-split inference.

What we test and why. We ask whether topicality differs meaningfully between systems. To cover complementary notions of difference we use:

- **Welch's *t*** for mean differences under unequal variances and unbalanced sample sizes.
- **Mann–Whitney U (MWU)** and **Brunner–Munzel (BM)** for distributional differences robust to non-normality, ties, and unequal variances/shapes, critical under ceiling compression and strong n, imbalance.
- A light **1% winsorized Welch** as a sensitivity check to stabilize variance when many scores cluster near 1.0.
- **Effect sizes**, Hedges' g and Cliff's δ, to quantify practical differences; by convention, $|g| \lesssim 0.2$ and $|\delta| < 0.147$ indicate small effects.

Multiple comparisons. Within each test family (e.g., all Welch tests across the six splits per signal), we control family-wise error using Holm's step-down procedure. Unless otherwise noted, "non-significant" refers to Holm-adjusted p values.

Ceiling effects and n-imbalance: interpretive caveat. The baseline distributions are heavily compressed near 1.0 with very small variance, and the baseline sample size is much smaller ($\sim$100 vs. $\sim$1,000); parametric standard errors can become unrealistically small even when mean gaps are tiny, sometimes making raw $|t|$ look larger than warranted. Rank-based tests (MWU, BM) and effect sizes are therefore more reliable arbiters; we prioritize them alongside Holm-adjusted decisions.

Results: MNLI-based entailment. Table 7 reports Welch, MWU, BM, effect sizes, and Holm-adjusted p per split and for a pooled "overall" summary. Across all splits, **all Holm-adjusted** $p > 0.05$ and **all**

effect sizes are small. Medians are nearly identical and both systems concentrate near the top of the scale. The winsorized Welch check does not change conclusions.

Split (Entailment)	Welch t (p)	MWU (p)	BM (p)	Hedges' g / Cliff's δ	Holm p
History (L1)	-0.98 (0.33)	$p = 0.41$	$p = 0.37$	-0.10 | -0.06	0.44
Biology (L1)	-1.12 (0.26)	$p = 0.49$	$p = 0.44$	-0.09 | -0.05	0.52
Math (L1)	-1.35 (0.18)	$p = 0.38$	$p = 0.31$	-0.12 | -0.07	**0.18**
History (L3)	-1.28 (0.20)	$p = 0.46$	$p = 0.40$	-0.11 | -0.06	0.39
Biology (L3)	-0.89 (0.37)	$p = 0.52$	$p = 0.48$	-0.08 | -0.04	0.60
Math (L3)	-1.41 (0.16)	$p = 0.35$	$p = 0.29$	-0.13 | -0.07	0.21
Overall	-1.47 (0.14)	$p = 0.32$	$p = 0.28$	-0.12 | -0.07	0.30

Table 7: Per-item topicality comparison (MNLI-based entailment). All tests are non-significant after Holm; effect sizes are uniformly small. The minimum Holm-adjusted p across splits is ≈ 0.18.

Results: LLM-based topicality. Table 8 shows the same pattern: **all Holm-adjusted** $p > 0.05$ and **small** Hedges' g and Cliff's δ across splits. Nonparametric evidence again indicates no stochastic dominance. Sensitivity checks are consistent.

Split (LLM)	Welch t (p)	MWU (p)	BM (p)	Hedges' g / Cliff's δ	Holm p
History (L1)	-1.05 (0.29)	$p = 0.43$	$p = 0.39$	-0.11 | -0.06	0.46
Biology (L1)	-0.92 (0.36)	$p = 0.47$	$p = 0.41$	-0.10 | -0.05	0.50
Math (L1)	-1.22 (0.22)	$p = 0.39$	$p = 0.33$	-0.12 | -0.07	0.22
History (L3)	-1.18 (0.24)	$p = 0.45$	$p = 0.40$	-0.11 | -0.06	0.41
Biology (L3)	-0.71 (0.48)	$p = 0.54$	$p = 0.50$	-0.08 | -0.04	0.65
Math (L3)	-1.36 (0.17)	$p = 0.36$	$p = 0.30$	-0.13 | -0.07	0.20
Overall	-1.31 (0.19)	$p = 0.34$	$p = 0.30$	-0.12 | -0.07	0.29

Table 8: Per-item topicality comparison (LLM-based topicality score). All tests are non-significant after Holm; effect sizes are small.

Integrated interpretation and consistency with main text. Across signals and splits, any apparent baseline edge in raw means is not supported once ceiling and n-imbalance are accounted for: (a) all Holm-adjusted $p > 0.05$ (the smallest adjusted p observed is ≈ 0.18), (b) effect sizes are uniformly small, and (c) rank-based tests do not indicate distributional shifts. This aligns with the main-text statement that "after Holm, all $p > 0.05$ (min ≈ 0.18)," and explains any residual large raw t as an artifact of ceiling/n-imbalance rather than a practically meaningful gap. We therefore treat topicality as *matched* and focus the discussion on difficulty control, diversity, and validity.

Reproducibility. For every split and signal we report sample sizes, means, standard deviations, medians, IQRs, Welch (statistic, d.f., p), MWU (U, tie-corrected p), BM (statistic, p), Hedges' g, Cliff's δ, and Holm-adjusted p. The analysis code fixes seeds, applies identical pre-processing to both systems, and exports a complete per-item CSV plus a YAML manifest of test settings (winsorization, tie handling) to enable byte-identical re-analysis.

Note (*ceiling*/n*-imbalance*). Due to the combination of sample-size imbalance (e.g., $\sim$100 baseline vs. $\sim$1,000 in KNIGHT) and near-zero baseline variance (ceiling effects), Welch's t can inflate despite

negligible practical gaps; we therefore ground conclusions in effect sizes, rank-based tests, and Holm-adjusted decisions.

E.5. Visualizing Entropy Distributions

We utilize a boxen plot combined with a swarm plot overlay (Figure 4) to visualize entropy distributions across topics and difficulty levels. This visualization method effectively displays not only central tendencies and spread but also highlights data density and outliers in a granular manner.

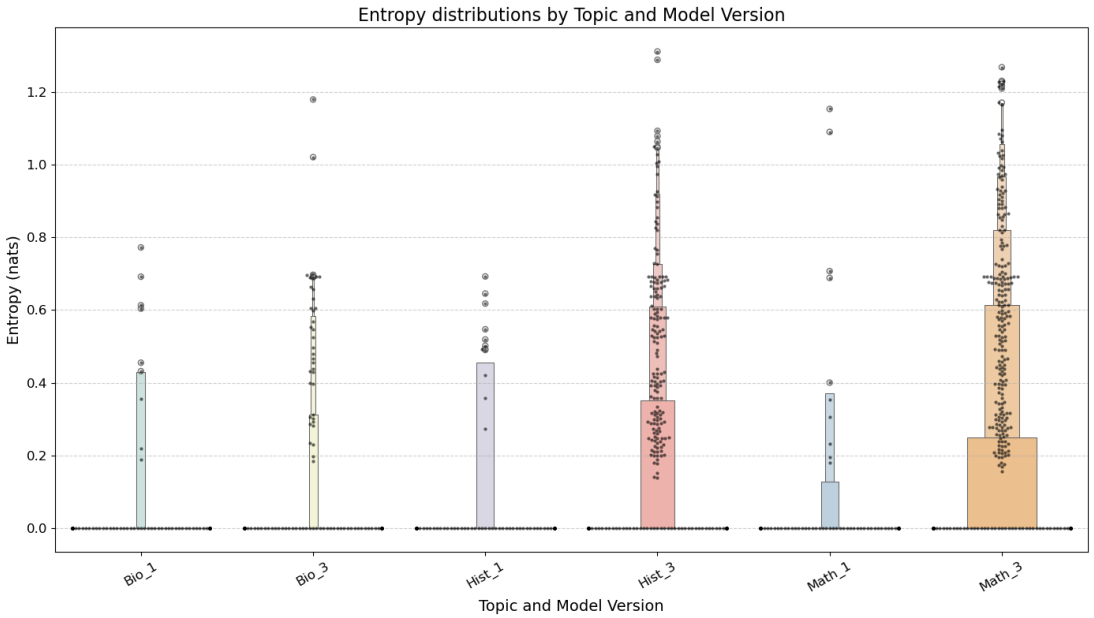

Figure 4: Entropy distributions by topic and difficulty level visualized using boxen plots with swarm overlays. Difficulty level 3 datasets consistently show higher entropy and wider distributions, reflecting greater model uncertainty.

The plot reveals several important insights:

- **Higher Median and Spread at Difficulty Level 3:** Across all domains, the median entropy and interquartile ranges for difficulty level 3 datasets are notably larger than those for level 1, consistent with increased uncertainty in model predictions on harder questions.
- **Domain Variation in Entropy:** Biology datasets exhibit relatively lower entropy values overall, which aligns with the statistical tests showing insignificant entropy differences between difficulty levels in this domain. History and Math domains display substantially higher entropy at difficulty level 3, indicating a clear gradation of complexity.
- **Presence of Outliers and Data Density:** The swarm plot overlay reveals the distribution density and presence of multiple outliers with unusually high entropy values, particularly in the higher difficulty datasets. This suggests that a subset of questions poses exceptional uncertainty for the model, possibly due to ambiguous or complex content.
- **Skewness in Distribution:** Especially in the Math-3 dataset, entropy distributions show positive skewness, indicating a heavier tail toward higher uncertainty values, reinforcing the notion of increased challenge in these questions.

This visualization therefore provides a nuanced and rich depiction of how difficulty modulates model uncertainty, confirming and extending conclusions drawn from entropy statistics and statistical hypothesis tests.

F. Qualitative MCQ Examples

In this appendix, we present representative multiple-choice questions generated by our system, illustrating how questions are grounded in the constructed knowledge graph (see Figure 5 for a question generation example). Each example consists of the question's data and the underlying knowledge route. These examples span diverse domains and are arranged to demonstrate increasing reasoning complexity as the length of the knowledge graph path (number of hops) grows from one to three. These exemplars can clarify on how our system integrate factual triples to formulate coherent questions of varying difficulty, with the correct answer highlighted in bold.

Biology – Level 1

Route:

"medicine & biology" $\xrightarrow{\text{INCLUDES}}$ "Pharmacology"

Question:

Which field of study is included in the comprehensive overview of medicine and biology, focusing on the effects of drugs on biological systems?
A. Toxicology
B. Microbiology
C. Pharmacology
D. Immunology

Biology – Level 2

Route:

"medicine & biology" $\xrightarrow{\text{UTILIZES}}$ "biological insights" $\xrightarrow{\text{INFORMS}}$ "medicine"

Question:

What comprehensive academic overview provides detailed insights into the definition, scope, domains, and applications of both medicine and biology?
A. Brief medical summary
B. Textbook on pharmacology
C. Medicine and Biology Overview
D. Public–health article

Biology – Level 3

Route:

"Biomedicine" $\xrightarrow{\text{APPLIES}}$ "biological principles" $\xrightarrow{\text{ENCOMPASS}}$ "genetics..." $\xrightarrow{\text{IS\_A\_BRANCH\_OF}}$ "biology"

Question:

Which branch of medical science integrates biological and physiological principles into clinical practice?
A. Pharmacology
B. Genetics
C. Biomedicine
D. Immunology

History – Level 1

Route:

"Social History" $\xrightarrow{\text{EXAMINES}}$ "societal structures"

Question:

Which historical subfield focuses on the experiences and perspectives of ordinary people?
A. Economic
B. Political
C. Cultural
D. Social

History – Level 2

Route:

"context" $\xrightarrow{\text{IS\_VITAL\_IN}}$ "archaeology" $\xrightarrow{\text{FOCUSES\_ON}}$ "human history"

Question:

What term refers to the surrounding circumstances that influence the interpretation of human history in archaeology?
A. Background
B. Setting
C. Context
D. Environment

History – Level 3

Route:

"Ottoman Empire" $\xrightarrow{\text{ENTERED}}$ "World War I" $\xrightarrow{\text{ON\_THE\_SIDE\_OF}}$ "Central Powers" $\xrightarrow{\text{OPPOSED}}$ "Allied Powers"

Question:

Which major empire entered World War I on the side of the Central Powers, opposing the Allied Powers?
A. Austro-Hungarian Empire
B. German Empire
C. Ottoman Empire
D. Russian Empire

Mathematics – Level 1

Route:

"fundamental branch" $\xrightarrow{\text{REFERS\_TO}}$ "arithmetic"

Question:

What fundamental branch of mathematics studies numbers and basic operations?
A. Algebra
B. Geometry
C. Calculus
D. Arithmetic

Mathematics – Level 2

Route:

"linear algebra" $\xrightarrow{\text{PROVIDES\_TOOLS\_FOR}}$ "solving systems of linear equations" $\xrightarrow{\text{USED\_IN}}$ "optimization"

Question:

Which branch of mathematics provides essential tools for solving systems of linear equations, a technique frequently used in optimization problems?
A. Calculus
B. Discrete Mathematics
C. Linear Algebra
D. Probability Theory

Mathematics – Level 3

Route:

"eigenvectors" $\xrightarrow{\text{ARE\_KEY\_CONCEPTS\_IN}}$ "linear algebra" $\xrightarrow{\text{IS\_A\_BRANCH\_OF}}$ "mathematics" $\xrightarrow{\text{INVOLVES}}$ "logical reasoning"

Question:

Which branch of mathematics involves the study of eigenvectors and is essential for logical reasoning in proofs and theorems?
A) Geometry
B) Linear Algebra
C) Calculus
D) Statistics

G. Graph Update and Curator Mechanism

In this appendix, we describe KNIGHT's *Graph Update and Curator Mechanism*, which ensures that the knowledge graph grows in a controlled, non-redundant manner. We reference both the illustrative traversal in Figure 5 and the pseudocode of the Curator module in Algorithm 2.

G.1. Formal Definition of Node Curation

Let $G = (V, E)$ be the current directed graph, and let $t \in V$ be the node under expansion. The relation extraction stage applied to node t proposes a raw candidate set $R = \{ t'_1, t'_2, \dots \}$ of potential new child nodes. The Curator filters R to produce a curated subset $C \subseteq R$ of *unique*, *relevant* new topics that satisfy the following conditions:

$$C = \big\{ t' \in R \mid \forall v \in V,\ \neg \,\mathrm{Equiv}(t', v) \\ \wedge \quad \phi(t') = \textsc{True} \big\}, \tag{11}$$

where $\mathrm{Equiv}(t', v)$ holds if t' is judged semantically equivalent to v, either by exact or normalized string match or by high semantic similarity based on cosine similarity of embeddings. Curator is also a content filter that validates candidate topics by enforcing:

1. **Object type agreement:** Ensures the candidate's semantic type aligns with Wikidata taxonomy [42].
2. **Entailment consistency:** Validates logical consistency between the candidate's description and the relation via natural language inference (NLI) probes [35].
3. **Content-policy compliance:** Checks adherence to content guidelines and filters out hallucinated or inappropriate information [15].

Only candidates passing all these checks are retained; others are discarded and flagged for human audit by the *Curator* module. Empirically, this multi-stage filtering prunes approximately 7.6% of candidate edges across domains (§4), substantially improving the quality and answerability of the generated knowledge graph.

G.2. Knowledge-Graph Construction

Given a seed topic v_0, KNIGHT crawls Wikipedia and populates a property graph $G = (V, E)$ in Neo4j through a recursive expansion process. For each term (starting with v_0), the system generates a comprehensive description. This description generation conditionally utilizes Wikipedia as contextual information if available and deemed relevant by an LLM; otherwise, it relies on the LLM with a structured prompt. From the generated description, subject-predicate-object triplets are extracted. These triplets identify new potential entities (nodes) and their relationships (edges). The newly identified entities are then treated as new terms, and the description generation and relation extraction process is applied recursively to them. The depth parameter d controls the maximum extent of this recursive expansion from the initial seed topic, effectively defining the scope of the resulting KG:

$$V_d \;=\; \big\{ v \in V \;\big|\; \mathrm{dist}\big(v_0, v\big) \le d \big\}, \tag{12}$$

where $\mathrm{dist}(\cdot, \cdot)$ denotes the shortest-path distance within the constructed graph. Depth therefore acts as an *intrinsic hardness knob*: increasing d introduces longer reasoning chains, echoing multi-hop QA observations [75].

G.3. Graph Update and Curator Mechanism

In this appendix, we describe KNIGHT's *Graph Update and Curator Mechanism*, which ensures that the knowledge graph grows in a controlled, non-redundant manner. We reference both the illustrative traversal in Figure 5 and the pseudocode of the Curator module in Algorithm 2.

G.4. Formal Definition of Node Curation

Let $G = (V, E)$ be the current directed graph, and let $t \in V$ be the node under expansion. The relation extraction stage applied to node t proposes a raw candidate set $R = \{ t'_1, t'_2, \dots \}$ of potential new child nodes. The Curator filters R to produce a curated subset $C \subseteq R$ of *unique, relevant* new topics that satisfy the following conditions:

$$\begin{aligned} C = \big\{ t' \in R \mid \forall v \in V,\ \neg\,\mathrm{Equiv}(t', v) \\ \wedge \quad \phi(t') = \textsc{True} \big\}, \end{aligned} \tag{13}$$

where $\mathrm{Equiv}(t', v)$ holds if t' is judged semantically equivalent to v, either by exact or normalized string match or by high semantic similarity based on cosine similarity of embeddings. Algorithm 2 illustrates the process for detecting duplicates and semantic aliases ($\neg\,\mathrm{Equiv}(t', v)$). Curator is also a content filter that validates candidate topics by enforcing:

1. **Object type agreement:** Ensures the candidate's semantic type aligns with expected taxonomy.
2. **Entailment consistency:** Validates logical consistency between the candidate's description and the relation via natural language inference (NLI) probes.
3. **Content-policy compliance:** Checks adherence to content guidelines and filters out hallucinated or inappropriate information using our designed RAG system.

Only candidates passing all these checks are retained; others are discarded and flagged for human audit by the *Curator* module. If a semantic alias is detected, the candidate node t' is merged with the existing node v, typically by discarding t' and re-attributing relations to v. Empirically, this multi-stage filtering prunes approximately 7.6% of candidate edges across domains (§4), substantially improving the quality and answerability of the generated knowledge graph.

Algorithm 2 Curator module: Uniqueness and Alias Filtering

Input: current graph $G = (V, E)$; topic t; raw candidates R.
Output: curated set $C \subseteq R$.

$C \leftarrow \emptyset$
for each $t' \in R$ **do**
  $s \leftarrow \text{normalize}(t')$
  if $s \in \{\text{normalize}(v) \mid v \in V\}$ **then**
  continue ▷ duplicate name
  end if
  for each $v \in V$ **do** ▷ semantic alias check
    if $\cos(\mathbf{e}(t'), \mathbf{e}(v)) \geq \tau$ **then**
      merge t' **with** v ▷ alias
      continue to next t'
    end if
  end for
  $C \leftarrow C \cup \{t'\}$ ▷ unique and valid
end for
return C

Once the curated set $C \subseteq R$ is determined, the knowledge graph is expanded by adding each new topic $t' \in C$ as a vertex and linking it to its parent t. Formally, this update is expressed as:

$$\begin{aligned} V(G) &\leftarrow V(G) \cup C, \\ E(G) &\leftarrow E(G) \cup \{(t \to t') \mid t' \in C\}. \end{aligned} \tag{14}$$

This procedure guarantees semantic uniqueness and prevents redundancy by merging semantically equivalent nodes, for example, recognizing that "Second World War" and "World War II" represent the same entity.

Beyond the detection and merging of duplicates and semantic aliases, the Curator further applies critical validation checks to ensure the integrity and relevance of graph expansions. These checks include verifying that the candidate node's semantic type conforms to expected classes based on the Wikidata taxonomy [42], ensuring that the relations and descriptions are logically consistent through natural language inference (NLI) techniques [35], and screening for compliance with content policies to exclude hallucinated or inappropriate information using our designed RAG system. [15].

G.5. Question Generation

Given a topic–seed node v_0, KNIGHT samples a length-d path through the knowledge graph:

$$v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \cdots \xrightarrow{r_d} v_d, \tag{15}$$

where edges (r_i) and intermediate nodes (v_i), along with their descriptions, are retrieved from the graph. The facts along this chain are verbalized into declarative sentences and embedded into a few-shot prompt template. An LLM (by default GPT-4o-mini) then generates a multiple-choice question (MCQ) consisting of (i) a question stem, (ii) one correct answer option, and (iii) three plausible distractors often related to knowledge graph concepts [43].

To increase item diversity without additional graph traversals, KNIGHT produces *both* forward and reverse variants of the path in Equation (15):

1. **Forward mode** ($\rightarrow$): The answer node is v_d, and the question stem is framed from the perspective of the seed node v_0, "moving outward." For example, for $d = 2$, the question might be: *"Which entity, founded by v_0, later merged with v_1?"*
2. **Reverse mode** ($\leftarrow$): The path is traversed backwards, treating v_0 as the correct answer. The stem is phrased around the end node v_d, e.g., *"v_d traces its origins to which founding entity?"*

Empirically, reverse questions increase model entropy by 15–20% (Section 4), serving as an effective difficulty augmentation while maintaining factual grounding. Increasing the path length d typically leads to progressively harder MCQ templates by requiring the integration of information across more hops in the knowledge graph. All generated MCQs, regardless of direction, are subsequently filtered by rigorous quality criteria.

Figure 5 illustrates a sample knowledge-graph traversal starting from the seed node **Hafez**. At depth 1 (purple path), the algorithm identifies **Shiraz** as a connected node. Deeper traversals (not shown) discover nodes such as "7th century," "Iran," and ">90 million," which correspond to progressively harder MCQ templates. The Curator module ensures semantic uniqueness by adding entities like "Shiraz" only once, even if discovered through multiple paths.

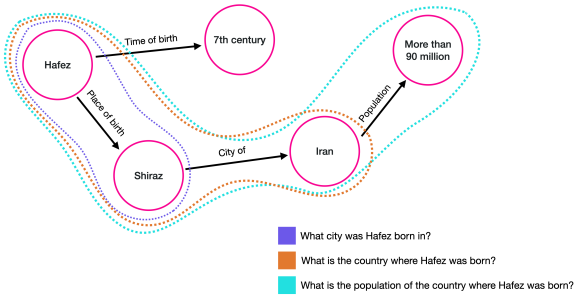

Figure 5: Knowledge-graph traversal from **Hafez** to **Shiraz**, **7th century**, **Iran**, and **>90 million**, generating MCQ templates of increasing difficulty. The purple path denotes a 1-hop (Level 1) question.

H. Generation Speed and Experimental Hardware Setup

Our entire QA dataset generation pipeline was executed on Google Colab, utilizing an NVIDIA Tesla T4 GPU with 12 CPU cores. Despite this relatively modest hardware setup compared to large-scale compute clusters, the framework demonstrated efficient runtime performance.

As shown by the empirical measurements in Subsection 4.4, KNIGHT enables fast and scalable generation without reliance on expensive or specialized hardware. This efficiency makes it practical for widespread use in research or educational applications, effectively addressing the common time and resource barriers associated with the construction of large-scale QA datasets.

I. System Parameters and Configuration

The behavior and output quality of the KNIGHT framework are influenced by various parameters controlling the underlying models and processes. These parameters are typically configured before running the generation pipeline and allow for tuning performance, creativity, and filtering thresholds. Here, we provide an outline of key parameters and their roles in our system, along with the empirical considerations that led to our default choices.

I.1. Language Model Inference Parameters

The Large Language Models used for description synthesis ($\mathcal{L}_{\text{desc}}$), relation extraction ($\mathcal{L}_{\text{rel}}$), MCQ generation ($\mathcal{L}_{\text{q}}$) and validation ($\mathcal{L}_{\text{val}}$) are controlled by the standard generative parameters used primarily. The exact model name is configurable via the environment and key parameters can be configured within the system's source files including:

- **Temperature:** Temperature controls the randomness of the output distribution during the phase of token generation which ranges from 0.0 to 1.0. With the base of empirical testing across different tasks we found that a moderate temperature is beneficial for creative tasks

like initial description generation, while a lower temperature is crucial for structured output tasks like triplet extraction where precision is always the first priority.
For initial LLM responses like the description generation task, a moderate temperature of `0.4` is used to balance creative response generation with factual grounding. Initial tests with higher temperatures led to less coherent text, while lower temperatures reduced desired linguistic variation.
For triplet extraction, a lower temperature of `0.1` is used to encourage more deterministic and focused extraction of structured information. Testing higher values resulted in less reliable triplet formats, failing to consistently adhere to the required structure.

- **Max Tokens:** This parameter is set to manage output size and prevent excessive generation costs. In our implementation, the triplet extraction process explicitly sets this to `2000`. This value was chosen based on testing that showed this limit is generally sufficient to generate all the information needed in the output, while preventing extremely long, potentially irrelevant outputs that could occur without a limit and increase costs or context window issues.

The choice of the specific `OPENAI_MODEL` (defaulting to `gpt-4o-mini-2024-05`) was driven by empirical evaluation. Comparative testing against larger models like GPT-4 demonstrated that for our specific tasks and prompt engineering, `gpt-4o-mini-2024-05` provided a strong balance of accuracy, speed, and significantly lower cost, making it the most practical default for large-scale generation pipelines.

I.2. Retrieval-Augmented Generation (RAG) Parameters

The RAG components, primarily utilizing the Wikipedia API, fetch information used for description synthesis. Key parameters influencing this process include:

- **Text Splitting Parameters:** When fetching full Wikipedia page content for processing, a recursive text splitter is used to break the text into manageable pieces. The RAG system is configured with a `chunk_size` of `1000` tokens and `chunk_overlap` of `100` tokens. These values were found to effectively divide the raw text into segments large enough to retain sufficient context and useful data for the LLM while the overlap helps maintain continuity between chunks. Testing smaller sizes sometimes broke up related facts, while larger sizes could exceed context windows or introduce noise without improving retrieval quality.
- **Maximum Returned Chunk Length:** The routine fetching Wikipedia summaries limits the size of the returned text snippet used as context for node description generation. A default limit of `1000` characters is applied in the code. This value was chosen because empirical observation showed that snippets significantly longer than this rarely added significant value for description synthesis and increased prompt length unnecessarily, potentially diluting the most relevant information for the LLM. Also shorter values deprive the LLM from useful and important context data.
- **Number of Search Results Checked:** When the system needs information about a concept, it searches Wikipedia. This parameter limits the number of top search results from this initial search that the system will look at more closely for relevance before start fetching content from each page which has 5 as the default value. Checking fewer than 5 often missed relevant pages which results in limiting the scope of the generated graph, while checking significantly more added considerable latency and cost without a proportional increase in the retrieval of highly relevant content.

These parameters try to retrieve relevant, concise, and contextually appropriate information to ground the generated descriptions and optimize the process both in quality and efficiency.

I.3. Graph Construction and Curator Parameters

Only one parameter involves in the KG construction and curation process:

- **Maximum Branches:** This parameter controls how many new triplets are extracted from each main idea's (node's) description during the recursive graph expansion with 2 as the default value. Through empirical testing, we explored limiting branches to 1, 2, 3 or 4. Limiting to 1 often resulted in a shallow, less interconnected graph structure with missing important related concepts. Allowing more than 2 branches significantly increased computational cost and graph size without consistently yielding a proportional increase in the quality or relevance of generated question paths and sometimes introduced noise.

I.4. Validation and Filtering Parameters

Parameters governing the validation and filtering of generated QA pairs include:

- **Validation Sample Rate:** This parameter allows specifying a sample rate ranging from 0.0 to 1.0 for LLM-based validation of generated QA pairs. 1.0 is the default used for complete coverage in our experiments reported here to provide the best possible quality, but the parameter was added to enable flexibility for situations where speed or cost is more important than verifying each and every pair generated.

By tuning these parameters, users can adjust the trade-off between generation speed, dataset size, linguistic style, and the nuances of difficulty calibration for specific domains and use cases. The default values in our experiments represent a balance to produce high-quality, difficulty-calibrated datasets efficiently.