Knowledge Graph Fusion with Large Language Models for Accurate, Explainable Manufacturing Process Planning

Danny Hoang[a], David Gorsich[b], Matthew Castanier[b] and Farhad Imani[a]

[a]School of Mechanical, Aerospace, and Manufacturing Engineering, University of Connecticut, Storrs, Connecticut, USA
[b]US Army DEVCOM GVSC, Warren, Michigan, USA

ARTICLE INFO

Keywords:
Large Language Model
Retrieval Augmented Generation
Knowledge Graph
Manufacturing Process Planning

ABSTRACT

Precision process planning in Computer Numerical Control (CNC) machining demands rapid, context-aware decisions on tool selection, feed-speed pairs, and multi-axis routing, placing immense cognitive and procedural burdens on engineers from design specification through final part inspection. Conventional rule-based computer-aided process planning and knowledge-engineering shells freeze domain know-how into static tables, which become limited when dealing with unseen topologies, novel material states, shifting cost-quality-sustainability weightings, or shop-floor constraints such as tool unavailability and energy caps. Large Language Models (LLMs) promise flexible, instruction-driven reasoning for tasks such as G-code synthesis to spindle-load queries, but they routinely hallucinate numeric values and provide no provenance. We present Augmented Retrieval Knowledge Network Enhanced Search & Synthesis (ARKNESS), the end-to-end framework that fuses zero-shot Knowledge Graph (KG) construction with retrieval-augmented generation to deliver verifiable, numerically exact answers for CNC process planning. ARKNESS (1) automatically distills heterogeneous machining documents, handbooks, G-code annotations, and vendor datasheets into augmented triple, multi-relational graphs without manual labeling, and (2) couples any on-prem LLM with a retriever that injects the minimal, evidence-linked subgraph needed to answer a query. Benchmarked on 280 industry-curated questions spanning tool sizing, feed-speed optimization, and tolerance diagnostics, ARKNESS provides uplifts up to +16.6 Percentage Point (pp) gain in multiple choice accuracy, +16.5 pp in F1-score, and 8.9× ROUGE-L on open-ended responses. Additionally, by grounding its reasoning in precise triples, ARKNESS lets smaller models match or surpass the accuracy of much larger cloud models, while running fully on-prem for privacy-preserving, real-time shop-floor inference.

1. Introduction

Modern production processes, such as precision machining, demands accuracy margins and tolerates virtually no numerical error, imposing substantial mental and operational loads on engineers from initial design specifications to the final part evaluation [1, 2, 3]. In high-stakes sectors, such as aerospace and energy, dimensional tolerances routinely tighten to $\pm$ 5 μm, while surface-integrity constraints, white-layer thickness, and residual stress, must remain within narrowly defined thresholds [4]. To meet these criteria, machinists must simultaneously optimize a variety of intertwined process parameters, such as tool type, feed rate, spindle speed, and depth of cut, carefully trading off cycle time against surface finish, tool wear, and geometric fidelity [5, 6, 7]. An incorrect decimal drill size or an outdated cutting-speed chart therefore translates directly into scrap, rework or latent defects that may only become apparent after catastrophic failure [8]. Such unplanned downtime erodes an estimated 11% of annual revenue for Fortune Global 500 manufacturers, roughly US $1.5 trillion lost each year [9].

Computer-Aided Process Planning (CAPP) engines and the Computer-Aided Manufacturing (CAM) modules bundled within mainstream Computer-Aided Design (CAD) suites (e.g., Siemens NX, CATIA, and SolidWorks) form automation pipeline that ingests STEP-compliant product models (AP203/242) and emits executable Numerical Control (NC) results [10]. A dedicated CAPP system executes the high-level analysis loop, geometry interrogation, feature recognition, precedence, constraint resolution, raw-stock/fixture/machine assignment, and operation-graph synthesis, before compiling feeds, speeds, and tool-change cycles into ISO6983 or STEP-NC. Embedded CAM modules then refine the middle layers: selecting cutters from vendor libraries, applying parameter presets, generating and simulating toolpath with collision checks and material-removal estimates, and invoking feature-based machining or knowledge-based-engineering rules for common prismatic features. Collectively, these rule-driven frame-

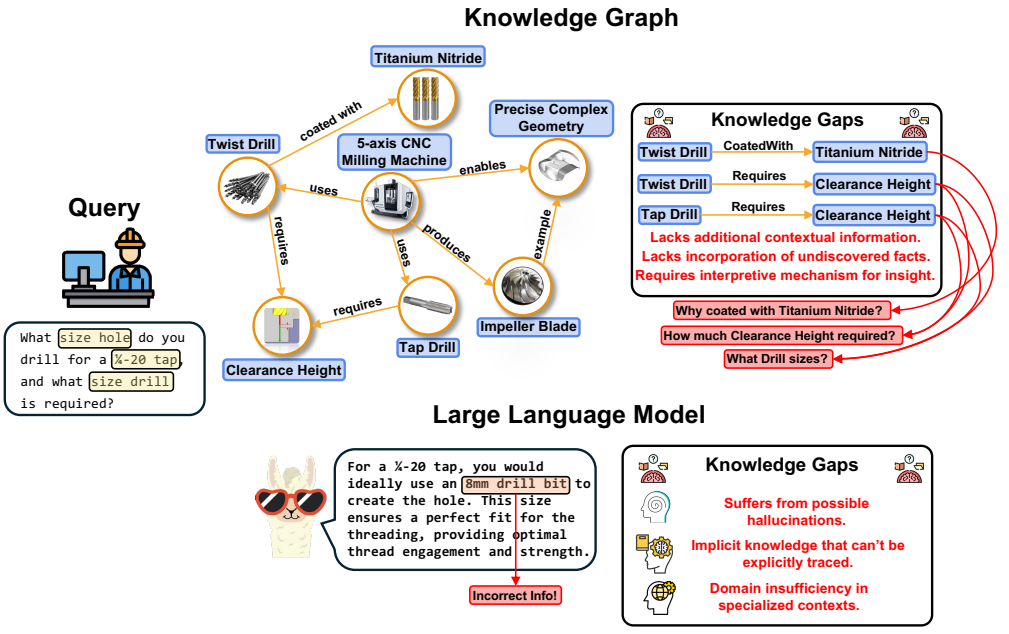

Figure 1: Illustration contrasting a knowledge graph and a large language model handling the same tap-drill sizing query.

works can compress planning lead time by 2–10×, yet they share a structural brittleness. Hard-coded heuristics falter when faced with unseen topologies (e.g., self-intersecting free-form pockets), novel material states, shifting cost–quality–sustainability weightings, or shop-floor constraints, such as tool unavailability and energy caps. Moreover, neither tier natively ingests telemetry (e.g., cutting force, vibration, and thermal imagery) generated by machining cells, forcing human experts to intervene, reconcile documentation, and manually retune parameters, an increasingly untenable bottleneck as geometrical complexity and throughput requirements escalate.

Large Language Models (LLMs) have emerged as versatile, instruction-following agents capable of synthesizing domain knowledge, reasoning under uncertainty, and generating executable results, with growing evidence of their benefits and limitations across operations, supply chain management, and digital manufacturing [11, 12, 13]. Recent work further shows that AI-LLM-driven analytics can support real-time decision-making in digital manufacturing by improving the scalability of optimization tasks such as demand forecasting, inventory control, energy-efficiency enhancement, and resource allocation [14]. Studies have also explored ChatGPT-enabled decision support for two-stage auctions in electric vehicle battery recycling and large language models for spare-parts demand forecasting, illustrating the broader shift from standalone automation toward LLM-assisted operational decision making [15, 16]. In manufacturing, studies demonstrate the ability of LLMs to synthesize 3-axis drilling paths, answer contextual queries about spindle loads and feed rates, and troubleshoot controller alarms with human-level accuracy in descriptive tasks [17]. Šket et al. benchmarked GPT-3.5 and GPT-4 across independent G-code generation, self-interpretation, and error simplification; GPT-4 rendered more correct toolpath yet remained limited to simple drilling operations and required extensive prompt steering [17]. To broaden functionality beyond code synthesis, ChatCNC was introduced, coupling multiple LLM agents with real-time machine telemetry so that an operator can, for example, ask "What was the spindle load at 09:32?"; the system attained 93.3% accuracy on complex production-tracking queries [18]. Nevertheless, both studies expose two systemic bottlenecks: first, dependence on precise user prompts or narrowly structured SQL retrieval, and second, fragile performance when required context is absent from the local database.

Domain-specific fine-tuning offers one mitigation strategy. Rosati et al. raised ROUGE-1 scores of Llama-3 from 0.164 to 0.314 using industrial user manuals [19]. CNCGPT fine-tuned GPT-3.5 Turbo on machine instructions and trouble-shooting logs, boosting factual correctness from ROUGE-L 0.296 to 0.692 [20]. More broadly, fine-tuned LLMs have also been explored for business optimization in digital manufacturing, highlighting their potential to support operational decision-making beyond narrow task-specific question answering [21]. Yet intensive fine-tuning and LLM-based automation present new trade-offs: (1) escalating computational cost as the query-response inventory grows into the thousands; (2) rapid obsolescence that mandates frequent retraining; and (3) data-sovereignty concerns when proprietary part files or production metrics traverse external cloud APIs, echoing broader questions about whether

LLMs should act as autonomous systems or decision support copilots in production and supply chain contexts [22]. Hence, a complementary mechanism is required, one that supplies rich, verifiable context to an LLM without chronically retraining the model or exposing sensitive data.

Knowledge Graphs (KGs), on the other hand, provide a structured, semantics-rich foundation for capturing machining know-how as typed entities (features, tools, operations, machines) and relations (e.g., requiresTool, precedes, causes) [23, 24, 25]. For CAPP, Wang et al. generated a feature-machining KG and proposed a cosine similarity retrieval that chose industry-valid hole-machining schemes with 0.8450 similarity to expert templates [26]. Guo et al. organized historical process routes by feature topology and machine capability, trimming part-routing time from 50–80 min to approximately 15 min [27]. Beyond planning, KGs have achieved >90% accuracy in root-cause diagnostics of rotary machinery even under missing sensor data by reasoning over causal chains [28]. Despite these gains, KG deployment still depends on laborious, manual curation: (1) domain experts must annotate source texts, validate triples, and periodically inject new machining knowledge; (2) Simple subject–predicate–object triples also struggle to encode context such as tolerance stack-ups, fixture constraints, or process-chain rationale for nuance components. For example, a given connection between the tool used to create thin walled geometry might omit the dependency between the optimal tool geometry corresponding the the feature size and specific tolerances required for that specific part.

Figure 1 showcases these limitations by contrasting (1) a standalone knowledge graph a standalone LLM. In the knowledge graph section, technical entities (i.e., Twist Drill, Clearance Height, Tap Drill, Titanium Nitride, 5-axis CNC Milling Machine, and Impeller Blade) are connected by explicit relations but can often be missing contextual links and undiscovered facts. The LLM section shows the model's generated answer ("8 mm drill bit"), annotated to expose imprecision risks, implicit untraceable knowledge, and domain inefficiency in specialized machining. Given these challenges, this research introduces Augmented Retrieval Knowledge Network Enhanced Search and Synthesis (ARKNESS), a hybrid framework that combines LLMs with semantically enriched, domain-specific KGs that provides answers that are grounded in validated information ensuring that recommendations for machining process planning queries are both reliable and contextual relevant. In addition, by grounding LLM prompts in these rich subgraphs, ARKNESS can deploy a compact parameter model on-premise with accuracy on par with, or exceeding, much larger cloud LLMs, slashing compute and deployment overhead. The main benefits of this paper are:

1. **A model-agnostic, KG-augmented CAPP assistant.** We present a retrieval-augmented generation pipeline that couples any large language model with a multi-relational machining knowledge graph. A graph-aware prompt compiler injects provenance-linked triples into the context window, yielding numerically precise and explainable process-planning answers.
2. **Self-supervised KG distillation from technical corpora.** A zero-shot, GPT-based entity–relation extractor converts heterogeneous machining documents (e.g., PDF manuals, specification sheets, and NC code comments) into contextualized triples, eliminating the manual curation bottleneck that has constrained previous CAPP-oriented KGs.
3. **Lightweight, on-prem hallucination suppression.** On a 280-query benchmark, ARKNESS paired with a smaller LLM model matches or surpasses much larger cloud LLMs while reducing hallucinations, demonstrating a privacy-preserving path to real-time shop-floor deployment.

The rest of the paper is as follows: Section 2 provides relevant knowledge graph and large language modeling research in manufacturing planning contexts, Section 3 provides the implementation of ARKNESS for automatic graph creation for retrieval and answering, Section 4 describes the experimental setup, Section 5 presents the experimental results, and Section 6 concludes the paper.

2. Research Background

2.1. Large Language Models in Machining

LLMs are rapidly transitioning from research curiosities to core enablers of AI-driven production, thanks to their capacity to parse free-form instructions, fuse heterogeneous context, and return actionable, domain-specific guidance. A recent survey by Li et al. categorized their early penetration into the manufacturing stack spanning generative CAD modeling, bio-process recipe design, robot path planning, and vision-based quality control [29]. Within subtractive manufacturing, the most direct application is natural-language-to-G-code translation. Šket et al. evaluated commercial ChatGPT models for 3-axis G-code generation testing GPT-3.5 and GPT-4 in three phases including independent G-code generation, interpretation of the generated G-code, and detecting and simplifying errors [17]. Their results

showed promise of implementing LLMs for G-code generation with GPT-4 producing more correct toolpaths but is severely limited to simple operations such as drilling. Additionally, their method relied heavily on user input to align the LLMs for generation which can lead to increased downtime if deployed in manufacturing environments. Jeon et al. expanded upon just G-code generation by developing ChatCNC that integrated various LLM agents with real-time CNC machining data [18]. This allowed users receive context-aware answers regarding the status of their 3-axis CNC machines such as spindle load at specific instances or times; their method achieved an accuracy of 93.3% in queries requiring complex data inference such as production tracking showcasing applications in analyzing data recorded in the manufacturing pipeline. Despite the high accuracy, the authors acknowledged that the model resulted in failures when encountered with missing context or the inability to retrieve information from their database. This suggests a more in depth search beyond conventional retrieval methods such as defined SQL database structures is required to supplement existing knowledge gaps.

Beyond prompting and traditional retrieving techniques, researchers have also introduced domain-specific finetuning to better align a LLMs responses with domain-specific knowledge. Rosati et al. finetuned Llama 3 for industrial applications achieving uplifts in average ROUGE-1 F1-score from 0.164 to 0.314 when trained on user manuals of a 360-degree camera [19]. Wang et al. finetuned GPT-3.5 for air craft maintenance outperforming general GPT-3.5 and its upgraded counterpart GPT-4.0 [30]. In areas specific to machining, Soundararajan et al. developed CNCGPT aimed for on-site CNC operator assistance by finetuning GPT-3.5 Turbo on machine-specific data, operational instructions, and troubleshooting assertions [20]. Results showed an increase in factual correctness when using ROUGE-L scores from 0.296 to 0.692 before and after finetuning, respectively. Despite the validity of their method, there are some challenges that might prevent further deployment on factory floors. One such issue is the amount of resources and training time required as the authors only trained around thirty query-response examples. In real world scenarios, there might be hundreds to thousands of scenarios and responses which can severely limit scalability during training. Moreover, as new scenarios continually emerge, the system must be frequently retrained to stay current which can further strain computational resources and complicates maintenance. There is also the challenge of security regarding production processes and information contained within. The authors used commercial LLMs which operate on external, cloud-based platforms where data flows and storage may raise concerns about data privacy and intellectual property protection. This reliance on third-party systems increases the risk of unauthorized access or data leakage, making it imperative to implement solutions that explore on-premise solutions to ensure that proprietary manufacturing data remains strictly confidential.

2.2. Knowledge Graphs in Machining

The ability to reuse and implement machining knowledge in a structured way provides not only enhanced consistency across planning and decision making but also ensures existing knowledge gaps are mitigated. Xiao et al. reviewed how computer-aided process planning involving knowledge graphs can benefit from reduced labor costs, shortened production cycles, and more intelligent use of existing information [23]. The authors analyzed key steps of implementing knowledge graphs from process knowledge representation to process knowledge graph construction and validation, showcasing how these methods can overcome traditional CAPP by reducing excessive manual interventions, and increasing flexibility and generalization. This implementation of process knowledge was shown in Wang et al. where they constructed a process knowledge graph for feature-based machining to automate machining scheme selection [26]. Through use of an improved cosine similarity formula for machining scheme selection, they achieved a similarity score of 0.8450 closely matching existing mature schemes implemented in industry for a typical shell part composed of 6 holes. By using their method, Wang et al. argued that the recommended machining steps would reduce tool load and increase both part quality and machining safety. In a similar study, Guo et al. created a knowledge graph for process route reuse by organizing historical process plans using part feature topology and machine capabilities [27]. When introducing a new part, the authors determine the process route that best aligns with its feature topology through a similarity check with existing process routes. Their case study on a shaft part resulted in completing the overall machining process route in about 15 minutes, down from the typical 50 to 80 minutes, significantly enhancing efficiency.

Another major application of machining knowledge graphs involves fault diagnosis and maintenance of CNC equipment. Manufacturing systems generate heterogeneous data from sources such as sensors, logs, and maintenance reports that if left in isolation or silos can be difficult for diagnosis if problems arise. Knowledge graphs offer a unified representation by connecting physical components, signals, and failure modes in a network of cause-effect and part-whole relationships. Qiu et al. tackled this "data island" problem by constructing a multi-layered knowledge graph that integrated sensor data and domain knowledge, enabling the automatic identification of health changes in the X-axis ball

screw drive system on a vertical milling tool (model XHK-5140) over 201 days, while similarity-based reasoning over the graph quantified the deviation from a healthy baseline [31]. Building on this idea, Cai et al. introduced a multi-level fault diagnosis KG for rotating machinery combining hierarchical knowledge of subsystem states with Bayesian reasoning, offering probabilistic inference across the graph to pinpoint root causes [28]. Notably, even with missing sensor outputs, their method achieved 91.1% diagnostic accuracy by leveraging the relationships between related symptoms and outperformed traditional rule-based fault diagnosis.

Despite the considerable progress in implementing knowledge graphs, there remain challenges in deploying them at scale for machining and industry. These aforementioned methods continue to depend heavily on extensive manual curation. The process of extracting, validating, and continuously updating the domain knowledge requires significant human knowledge and intervention, making it difficult to scale rapidly or adapt quickly to new information. This labor-intensive approach not only increases the risk of inconsistencies or omissions but also impedes real-time responsiveness in dynamic manufacturing environments. Furthermore, the conventional reliance on simple triple-based representations constrains the ability to capture the full contextual richness necessary for nuanced decision-making in machining processes. Such representations often fall short in encoding the detailed circumstances, rationale, and intricate relationships that underlie complex manufacturing operations. As a result, they may fail to fully express how various machining operations interrelate or why specific process choices are chosen. This would thus lead to a superficial understanding of the underlying mechanics and process dependencies. These limitations highlight the need for more flexible and richly contextualized approaches in knowledge representation to support the evolving demands of modern machining applications.

2.3. Motivation

The studies reviewed above demonstrate that LLMs can meaningfully assist machining workflows, but they also expose recurring limitations that are difficult to ignore in CNC process planning. For example, LLM-driven natural-language-to-G-code generation shows promise, yet it remains constrained to relatively simple operations and can require substantial user intervention to obtain usable toolpaths, which is not well aligned with high-throughput shop-floor settings [17]. Likewise, agentic assistants that integrate real-time CNC data can achieve high accuracy on specific query types, but they can fail when critical context is missing or when the system cannot retrieve the required information from its database [18]. Domain-specific fine-tuning improves apparent factual correctness in targeted settings, but it introduces practical deployment bottlenecks, including the need for continual retraining as scenarios evolve and heightened concerns around confidentiality when relying on commercial cloud-based models for proprietary manufacturing knowledge [19, 30, 20]. Collectively, these results motivate a solution that maintains the flexibility of LLM interaction while providing stronger guarantees for provenance, numerical consistency, and deployment constraints that matter in practice.

While LLM-centric approaches emphasize interaction and broad coverage, Table 1 highlights that they typically do not provide provenance-linked outputs or robust numerical behavior when the relevant context is incomplete. In parallel, machining knowledge graphs offer structure for reuse and constrained reasoning, but current approaches often depend on labor-intensive curation and, when represented as isolated triples, can lose the contextual conditions and validity constraints required for engineering justification [23, 26, 27, 31, 28]. This creates a persistent gap between usability and trustworthiness, particularly for numerically sensitive decisions where answers must be auditable and locally deployable.

ARKNESS closes these limitations through an end-to-end hybrid pipeline that automatically distills heterogeneous machining documents into contextualized graph knowledge with paragraph-level provenance, retrieves a compact evidence subgraph tailored to the user query using vector-embedding search combined with beam search, and conditions a downstream LLM on this evidence to produce numerically grounded and auditable answers. To further prioritize correctness in deployment, ARKNESS incorporates conservative verification and a human-in-the-loop auditing design that focuses manual effort on flagged or ambiguous cases rather than requiring exhaustive annotation.

3. Research Methodology

This section details the two main components of the ARKNESS framework with 1) Knowledge graph construction and 2) Knowledge graph inference. Figure 2 provides the general overview of the two components.

Table 1
Comparison of representative machining LLM and KG approaches.

Approach	Studies	Grounded and Traceable	Numeric Reliability	Scalable Updates	On-prem Suitable
LLM-only machining assistants	Šket et al. [17]; Jeon et al. [18]	✗	✗	✓	✗
LLM fine-tuning with open-weight models	Rosati et al. [19]	✗	✗	✗	✓
LLM fine-tuning with proprietary API models	Wang et al. [30]; Soundararajan et al. [20]	✗	✗	✗	✗
Machining knowledge graphs for planning and reuse	Xiao et al. [23]; Wang et al. [26]; Guo et al. [27]	✗	✓	✗	✓
Machining knowledge graphs for diagnosis and maintenance	Qiu et al. [31]; Cai et al. [28]	✗	✓	✗	✓
ARKNESS (ours)	This paper	✓	✓	✓	✓

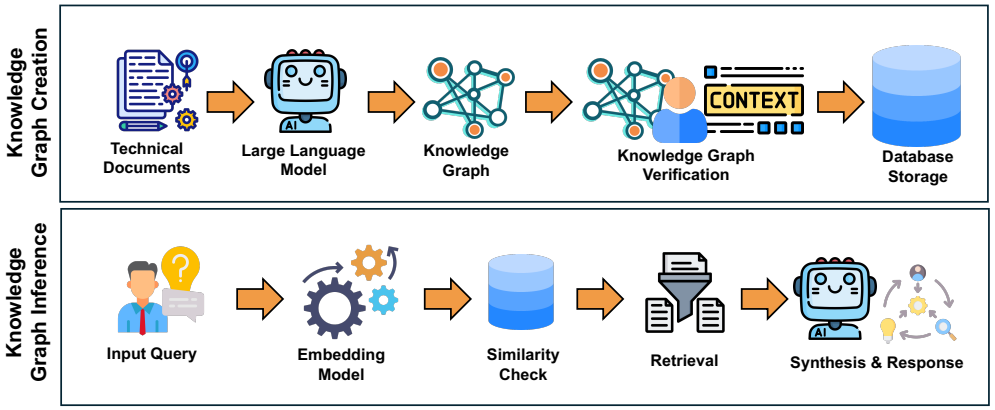

Figure 2: Overview of the framework from graph construction and storage through semantic search, matching, and retrieval to the final model response.

3.1. Knowledge Graph

A knowledge graph can be considered as a set of tuples $\mathcal{G} = \{(s, r, o)\}$ where v and w are vertices (entities) from a set of vertices $\mathcal{V}$ and r is an edge (relationship) from a set of relationships $\mathcal{R}$. The set of tuples is commonly known as triples denoted as (*head*, *relation*, *tail*) representing s, r, and o, respectively.

3.2. Automated Graph Construction

The knowledge graph construction for ARKNESS is mainly done using textual data from documents relevant to the users chosen domain. Due to the heterogeneous nature of document types and file formats (e.g., .docx, .pdf, .pptx), preprocessing was first done to extract the textual information from these different sources. This was achieved through the Docling [32] Python package which efficiently parses through the documents and extracts the raw text subsequently exporting the content as Markdown files for further processing.

Document splitting was done through parsing out each individual paragraph. Given a document with p individual paragraphs, each paragraph was given to a LLM to extract entities and relations forming T triples. Specific instructions were given to guide the LLM to output structured information given below:

```
-Goal-
Extract structured triples directly from the input text in the following format:
```

```
ENTITY_1, RELATIONSHIP_TYPE, ENTITY_2, "RELATIONSHIP_DESCRIPTION"

-Steps-
1. Read the input text carefully to identify:
   - Key entities: Concepts, systems, technologies, or processes central to the text.
   - Relationships: Clear actions or connections described in the text that link two
entities.
   - Descriptions: Verbatim or paraphrased descriptions from the text that explain the
relationship.

2. For each relationship, construct a triple:
   - ENTITY_1: The primary concept or entity initiating the relationship.
   - RELATIONSHIP_TYPE: The action or connection type as described in the text.
   - ENTITY_2: The target concept or entity affected by ENTITY_1.
   - RELATIONSHIP_DESCRIPTION: A concise description of the relationship directly sourced
 from the input text.

3. Each triple must be clear and in this format:
   ENTITY_1, RELATIONSHIP_TYPE, ENTITY_2, "RELATIONSHIP_DESCRIPTION"

4. Use the original text verbatim where possible for the description, ensuring accuracy.
Avoid adding external interpretations or explanations.
```

This process was repeated for each document chosen creating G_D subgraphs. The purpose of including the 'Relationship Description' in the triple information is to retain the relevant contextual information by which the triple was created. This removes ambiguity and often missing information found in traditional knowledge graphs. All subgraphs were then stored into respective text files which were then combined together for further processing.

A relational database was then created using PostgreSQL to efficiently store and retrieve entities, relations, and their associated context. Specifically, Python and the Psycopg library was used to interface with the database where during initialization three primary tables are created: (1) subjects table: stores unique entity names with a serial primary key, (2) relations table: associates each subject with its corresponding relationships where each entry references a subject's unique identifier, ensuring relationships are correctly linked, and (3) objects table: stores ending triple entities along with corresponding contextual information. This structure can be also represented as follows. Let S be the set of subjects (entities). For each subject $s \in S$, let $R(s)$ be the set of relations associated with s. For each relation $r \in R(s)$, let $O(r)$ be the set of ending objects linked to r, where each object is represented as a tuple (o, c) with o representing connected entities and c the corresponding context or provenance. The entire data base is thus defined by the set:

$$\mathcal{D} = \{\langle s, r, (o, c)\rangle \mid s \in S,\ r \in R(s),\ (o, c) \in O(r)\} \tag{1}$$

with a single triple entry represented as:

$$\langle s, r, (o, c)\rangle \tag{2}$$

3.3. Triple Verification

After constructing candidate triples, we apply a natural language inference (NLI) verification pass. Let $\mathcal{T}$ denote the set of candidate triples, where each $t = \langle s, r, (o, c)\rangle$ consists of a subject s, relation r, object o, and supporting context c originating from the source text. We deterministically verbalize each triple to a hypothesis via a linearization function $\ell : \mathcal{T} \to \mathcal{H}$ and set $h := \ell(t)$. From the provenance c, the implementation selects a single premise sentence $p \in c$ to serve as the textual evidence for verification. We verify each candidate triple using a single-premise natural language inference pass that mirrors the symbols introduced in Section 3.2. For a triple $t = \langle s, r, (o, c)\rangle$, we first form

a hypothesis by linearizing the triple and we select one premise sentence from its provenance:

$$h := \ell(t), \qquad p \in c \tag{3}$$

The ordered pair (p, h) is scored by a pretrained cross-encoder NLI model M, which returns three real-valued logits in the fixed label order [contra, neutral, entail]. Applying a softmax yields calibrated probabilities:

$$\big(P_C(p,h),\ P_N(p,h),\ P_E(p,h)\big), \qquad P_C + P_N + P_E = 1, \tag{4}$$

interpreted as the model's belief that the premise contradicts, is neutral with respect to, or entails the hypothesis, respectively. We map these probabilities to a discrete verification decision using a conservative, two-threshold gate. Let $\tau_E, \tau_C \in (0, 1)$ denote the entailment and contradiction thresholds (defaults being $\tau_E = \tau_C = 0.60$). The decision function $g : \mathcal{T} \to \{\text{entailed}, \text{neutral}, \text{contradicted}\}$ is

$$g(t) = \begin{cases} \text{entailed}, & \text{if } P_E(p,h) \geq \tau_E \text{ and } P_E(p,h) \geq P_C(p,h), \\ \text{contradicted}, & \text{if } P_C(p,h) \geq \tau_C \text{ and } P_C(p,h) \geq P_E(p,h), \\ \text{neutral}, & \text{otherwise.} \end{cases} \tag{5}$$

Thus, acceptance requires both a minimum level of entailment and that contradiction is not more probable than entailment; contradiction is asserted only when it is simultaneously above threshold and not weaker than entailment. All remaining cases, including high neutral mass or sub-threshold ties, are assigned to the neutral outcome. From these scores, Triples classified as contradictory or neutral undergo manual review and, where needed, revision prior to inclusion in the final knowledge graph; triples classified as entailed are added directly.

3.4. Graph Retrieval and Traversal

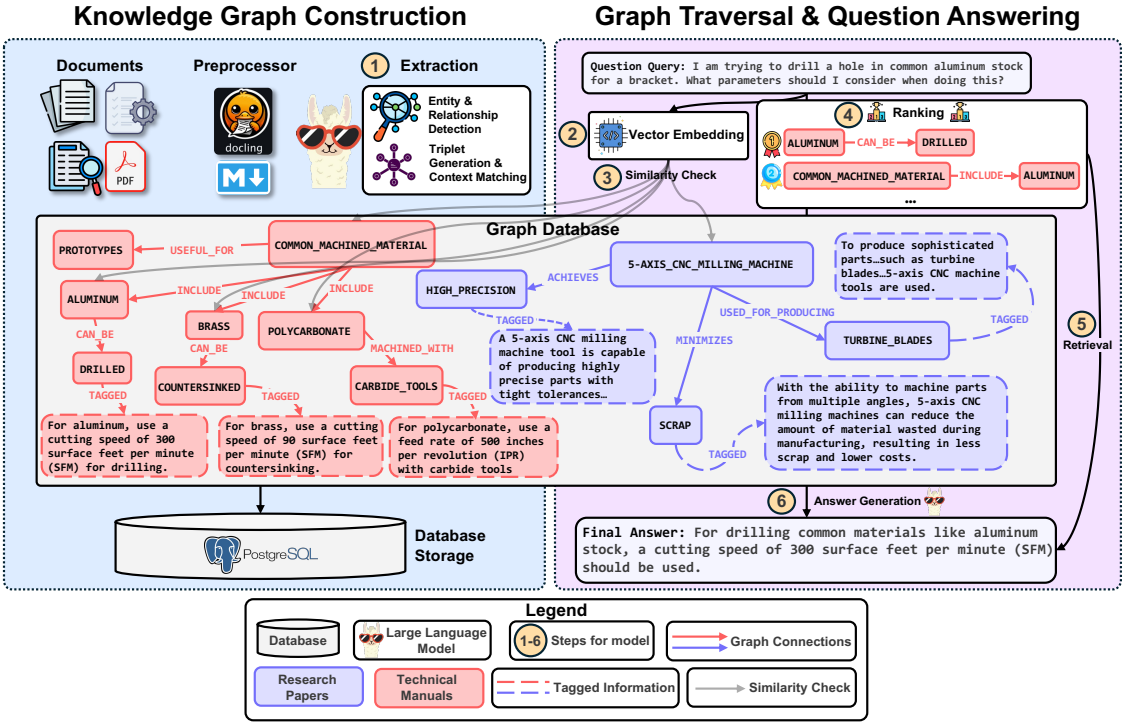

Figure 3: Overview of the knowledge graph construction and graph traversal for user queries.

Figure 3 provides an in-depth illustration of how the knowledge graph triples are compared against a query and

retrieved to be given to a LLM. The left side depicts the automated graph construction process from document selection, preprocessing, entity-relationship extraction, triple generation and context matching, and database storage. The right side depicts knowledge graph retrieval given a user query by embedding the query to get a vector representation, performing a similarity check over the triples in the graph, and ranking the most relevant triples to give to the large language model for answer generation. The retrieval process begins by first encoding a given natural language query, or question, into a vector $q \in \mathbb{R}^d$. This is achieved using a chosen semantic embedding model $f : \text{Text} \to \mathbb{R}^d$. Similarly, each triple t_i from the database is encoded into embedding e_i using the same semantic embedding model where $f : t_i \to \mathbb{R}^d$. The similarity between the query and each triple is then calculated using cosine similarity according to:

$$\cos(q, e_i) = \frac{q \cdot e_i}{\|q\| \|e_i\|} \tag{6}$$

This similarity score provides an evaluation of how semantically similar the query given is to each of triple in the knowledge graph allow one to ascertain its relevance.

Given the potentially massive scale of knowledge graphs, which can reach hundreds of thousands or more connections, we replace fixed hyperparameters with data-adaptive rules while preserving the same retrieval flow and level of interpretability. Let the embedded query be $q \in \mathbb{R}^d$. Each triple is represented as $t_i = (s_i, r_i, o_i, c_i)$, where s_i, r_i, o_i denote the subject, relation, and object strings and c_i denotes an associated textual context. We embed each triple in two ways: a "triple" embedding $v_i \in \mathbb{R}^d$ that summarizes (s_i, r_i, o_i, c_i) and a "context" embedding $u_i \in \mathbb{R}^d$ that summarizes c_i alone. We first measure base and context similarities to the query and combine them into a refined score so that the ranking reflects both structural and contextual alignment:

$$s_i^{\text{base}} = \cos(q, v_i), \qquad s_i^{\text{ctx}} = \cos(q, u_i), \qquad \tilde{s}_i \;=\; s_i^{\text{base}} + \lambda\, s_i^{\text{ctx}}, \quad \lambda = \tfrac{1}{2}. \tag{7}$$

Having only the highest refined similarities chosen allows the most semantically relevant triples to be selected for further processing. To limit computation while preserving recall, we form a pre-pool using only the base similarity, keeping either a chosen fixed minimum or a constant fraction of the graph, whichever is larger:

$$P_{\text{size}} \;=\; \min\big(N,\ \max(M_{\min},\ \lfloor \alpha N \rfloor)\big), \qquad \mathcal{P} \;=\; \text{Top-}P_{\text{size}}\Big(\{\, s_i^{\text{base}} \,\}_{i=1}^{N}\Big), \tag{8}$$

where N is the number of triples, $\alpha \in (0, 1]$ is a pruning fraction (e.g., $\alpha = 0.2$), and $M_{\min}$ is a small absolute minimum (e.g., $M_{\min} = 100$). Over the refined scores restricted to $\mathcal{P}$, we compute a location and scale estimate and define a data-adaptive inclusion floor via a z-threshold z (e.g., $z = 0.5$). This produces a dynamic choice of K that automatically adapts to the score distribution while remaining within safe bounds:

$$\mu \;=\; \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \tilde{s}_i, \qquad \sigma \;=\; \sqrt{\frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} (\tilde{s}_i - \mu)^2}, \qquad \tau \;=\; \mu + z\,\sigma, \tag{9}$$

$$\widehat{K} \;=\; \Big|\{\, i \in \mathcal{P} : \tilde{s}_i \geq \tau \,\}\Big|, \qquad K \;=\; \begin{cases} \min\big(K_{\max},\ \max(K_{\min},\ \widehat{K})\big), & \widehat{K} > 0, \\ K_{\text{base}}, & \text{otherwise}, \end{cases} \tag{10}$$

where $K_{\min}$ and $K_{\max}$ bound the pool size (e.g., $K_{\min} = 5$, $K_{\max} = 50$) and K_{base} is a conservative fallback (e.g., $K_{\text{base}} = 10$). The dynamic candidate pool is then obtained by sorting the refined scores and retaining the top indices:

$$C_0 \;=\; \{\, i \in \mathcal{P} \mid i \text{ ranks among the Top-}K \text{ of } \{\tilde{s}_i : i \in \mathcal{P}\} \,\} \tag{11}$$

As mentioned previously, because a knowledge graph can contain hundreds of thousands or even millions of connections, relying solely on the top-K triples could overlook additional relevant triples connected to them. To explore the extended neighborhood of these triples, we traverse the graph by performing a beam search expansion with a depth limit d_{max}. Let $E(i)$ denote the set of adjacent embedded triples indexed by i (i.e., those sharing a node with triple i).

Each neighbor $j \in E(i)$ is scored using the same refined rule to maintain consistency between selection and expansion, and we also propagate the global floor into the traversal stage in the form of a *node-level* pruning condition (i.e., a parent is skipped if all of its neighbors are weak):

$$r_j \;=\; \cos(q, v_j) + \lambda\, \cos(q, u_j), \qquad \tau_{\text{exp}} \;=\; \eta\, \tau, \;\; \eta = 1.0, \qquad \text{skip node } i \text{ if } \max_{j \in E(i)} r_j \;<\; \tau_{\text{exp}} \tag{12}$$

For the initial depth $d = 0$, we set the initial candidates as C_0. For each subsequent depth $d \geq 1$, we select neighbors using a *dynamic* (node-specific) beam width that adapts to the local score distribution: for each parent $i \in C_{d-1}$, we compute the local mean score over its neighbors and retain as many neighbors as score at least this mean, capped by a beam budget $B_{\max}$. This widens the beam when evidence is strong and narrows it when evidence is weak:

$$\bar{r}_i \;=\; \frac{1}{|E(i)|} \sum_{j \in E(i)} r_j, \qquad b(i) \;=\; \min\Big(B_{\max},\, \max\big(1,\, \big|\{\, j \in E(i) : r_j \geq \bar{r}_i \,\}\big|\big)\Big), \;\; B_{\max} = 10, \tag{13}$$

and we then select the top $b(i)$ neighbors by score without applying an additional per-edge threshold:

$$C_d \;=\; \bigcup_{i \in C_{d-1}} \text{Top-}b(i)\Big(\{\, j \in E(i) \,\}\Big), \qquad d = 1, 2, \ldots, d_{\max}, \qquad d_{max} = 10 \tag{14}$$

In order to avoid re-processing triples and to prevent forward/backward duplicates during traversal, we canonicalize edges to a forward orientation and maintain a visited set over *contexted* canonical triples, i.e., keys of the form (s, r, o, c). Only edges whose canonical key is unseen are considered, and the key is inserted into the visited set immediately upon selection, yielding the visited-constrained update:

$$V \subseteq \{(s, r, o, c)\}, \qquad C_{d+1} \;=\; \bigcup_{i \in C_d \setminus V} \text{Top-}b(i)\Big(\{\, j \in E(i) \,\}\Big) \tag{15}$$

After recursively expanding through the chosen maximum depth and dynamic beam sizes, the final set of retrieved triples is defined by

$$C^* \;=\; \bigcup_{d=0}^{d_{\max}} C_d. \tag{16}$$

As a final step, after the final set of candidate triples is chosen, the corresponding context c_i for each selected triple is retrieved from the database. This creates a set $\mathcal{C}$ of relevant information ready for downstream use:

$$\mathcal{C} \;=\; \Big\{\, \langle s_i, r_i, (o_i, c_i)\rangle \;:\; i \in C^* \,\Big\} \tag{17}$$

3.5. Large Language Model Generation

The last step of answering the query q involves prompt construction where the information retrieved from the knowledge graph is given to the large language model to help its answering. This is done by defining a set of system instructions that designate that the LLM is designed to answer questions incorporate both its own knowledge and the given retrieved knowledge. For example, we define the system instructions as:

$$I_{\text{sys}} = \begin{array}{l}\text{'You are designed to help answer questions using retrieved knowledge. Not all knowledge given}\\ \text{need to be used but focus on the most important information. Remember this knowledge, if}\\ \text{there is any, to help your decision making:'}\end{array} \tag{18}$$

Then the final prompt P given to the LLM is constructed by concatenating the system instructions, query, and retrieved knowledge graph information:

$$P(q, \mathcal{C}) = I_{\text{sys}} \oplus \mathcal{C} \oplus q \tag{19}$$

The final answer A_{LLM} is then generated by the LLM:

$$A_{\text{LLM}} = \text{LLM}\big(P(q, C)\big) \tag{20}$$

4. Experimental Design

4.1. Knowledge Graph

To test the efficacy of ARKNESS, we first created our machining knowledge graph composed of research and technical documents spanning 5-axis milling capabilities, industrial robotic integration in CNC machining, sustainability in CNC machining, CNC programming and process planning, and fundamentals of CNC machining. By exposing ARKNESS to a spectrum of complexity levels, domain-specific terminologies, and the assembled documents enables a comprehensive evaluation of the model's CNC machining knowledge. Each document was processed using the automated graph-construction pipeline detailed in Section 3.2 using GPT-4o [33]. Descriptions of each documents chosen are listed in Table 2 along with the total number of entities, relations, and triples after processing. In total, the generated knowledge graph contains 4329 triples, 6659 unique entities, and 1251 unique relations providing a relatively large graph to test to test ARKNESS' capacity robust inference, and accurate knowledge retrieval across diverse CNC-machining scenarios.

To assess and control the accuracy of LLM-generated triples during knowledge graph construction, we implemented the conservative verification pipeline with full provenance described in Section 3.3. Each candidate triple is converted into a simple hypothesis sentence (e.g., (5 AXIS_CNC_MILLING_MACHINE, used_for_producing, TURBINE_BLADES) becomes "5 AXIS_CNC_MILLING_MACHINE used_for_producing TURBINE_BLADES") and checked against its source context using the RoBERTa-MNLI natural language inference model [43]. The model produces entailment, neutral, and contradiction scores, which we convert into a conservative gate where a triple is automatically accepted only when entailment exceeds a threshold of 0.6 and dominates contradiction; otherwise, it is flagged as neutral or contradicted according to Equation 5. Across our corpus, the NLI gate labeled 4329 candidate triples as 3158 accepted (72.9%), 69 contradicted (1.6%), and 1102 neutral (25.5%). We manually reviewed the neutral and contradicted groups, corrected errors where needed, and only then included validated triples in the final graph. As a result, the final KG contains only triples that are either directly supported by the source according to the NLI model or confirmed through manual validation, providing strong source alignment and reducing the need for large-scale post hoc cleaning.

As an additional validation step, we conducted a quantitative evaluation of triple extraction quality using small-scale sampling and gold annotation. We first estimated precision through manual verification of 400 randomly sampled triples from the KG, stratified by each document's KG size. For each sampled triple, we inspected the (subject, relation, object) statement together with its source context and labeled it as correct if it was directly supported by the text, and incorrect otherwise. As shown in Table 3, this yields a precision of 0.845 (338/400) with a 95% confidence interval of [0.806, 0.877], indicating that most extracted triples are factually supported by their provenance. To estimate recall, we constructed a small gold subset by manually annotating triples for 100 randomly sampled source contexts, yielding 161 gold triples. We then ran GPT-4o on the same contexts and matched extracted triples to gold using exact match and semantic similarity to accommodate minor surface-form differences in entities and relations, with manual checks for ambiguous cases. A gold triple was counted as recovered if an extracted triple expressed the same underlying fact, even if the wording differed slightly. This yields a recall of 0.714 with a 95% confidence interval of [0.64, 0.778]. For completeness, we report the corresponding F1 computed from these estimates (0.774), noting that precision and recall are computed on different evaluation subsets (random extracted-triple sample vs. gold-annotated subset), which is standard in sampling-based validation when large-scale annotation is infeasible. Taken together, these results suggest that GPT-4o produces largely correct triples with substantial coverage, while our conservative verification and selective manual review steps provide an additional safeguard. In practice, this human-in-the-loop design substantially accelerates KG construction by focusing manual effort on a small set of flagged or ambiguous cases rather than requiring exhaustive annotation.

4.2. Models

To rigorously assess how model choice shapes our framework's performance, we evaluate a diverse set of both open-source and closed-source large language models across multiple parameter scales. For open sources models in their 8 bit configuration, Llama 3.2 3B Instruct [44], LLama 3.1 8B Instruct [45], and Qwen 2.5 7B Instruct [46] were

Table 2
Overview of source documents and their corresponding knowledge graph sizes.

Document	Description	Knowledge Graph Size
A Review in Capabilities and Challenges of 5-Axis CNC Milling Machine Tool Operations [34]	This review surveys recent advances and challenges in 5-axis CNC milling, including error modeling and compensation, toolpath and process optimization, virtual machining systems, tool wear and temperature prediction, and sustainability considerations.	# of triples: 464 # of unique entities: 654 # of unique relations: 199
Design and development of a CNC machining process knowledge base using cloud technology [35]	This paper presents a cloud-based CNC machining process knowledge base that maps STEP-NC to an OWL ontology and leverages Hadoop's HBase for scalable storage, MapReduce driven querying, and SWRL-based reasoning to enable intelligent, high-throughput process planning.	# of triples: 559 # of unique entities: 828 # of unique relations: 291
Exploring the Application of Industrial Robots in CNC Machining [36]	This paper explores the deployment of industrial robots in CNC machining. It details robot selection, workflow and control program design, joint and transmission mechanisms, and drive system choices to automate loading/unloading and enhance operational flexibility, precision, and efficiency.	# of triples: 98 # of unique entities: 151 # of unique relations: 57
Fundamentals of CNC Machining [37]	This guide provides a practical introduction to CNC machining fundamentals covering shop safety, tooling, coordinate systems, programming and operation of mills and lathes, 2D/3D toolpaths, workholding examples, and best practices for prototype and short-run production.	# of triples: 1902 # of unique entities: 2966 # of unique relations: 512
Innovative Approaches to Sustainable CNC Machining: A Machine Learning Perspective on Energy Optimization [38]	This paper develops a machine-learning framework for sustainable 5-axis CNC milling by combining per-axis power monitoring with Taguchi experimental design and tree-based regression to model and predict energy consumption.	# of triples: 278 # of unique entities: 452 # of unique relations: 183
Integration of Taguchi and PROMETHEE for CNC Milling Machining Parameter Optimization [39]	This paper integrates Taguchi's orthogonal-array experimental design with the PROMETHEE multi-criteria ranking method to optimize spindle speed, feed rate, and depth of cut for CNC milling of AA6061.	# of triples: 227 # of unique entities: 366 # of unique relations: 74
Research on CNC programming and machining process based on CAD/CAM technology [40]	This paper introduces a CAD/CAM-based CNC programming framework that uses Bayesian process-skeleton mapping to link part features with reusable macro-processes and applies residual-height trajectory generation and nonlinear-error compensation to optimize multiaxis toolpaths.	# of triples: 269 # of unique entities: 443 # of unique relations: 167
Review on Design Research in CNC Machine Tools Based on Energy Consumption [41]	This review surveys global advances in modeling, designing, and evaluating CNC machine tools from an energy-consumption perspective.	# of triples: 270 # of unique entities: 470 # of unique relations: 140
Robotical Automation in CNC Machine Tools, A Review [42]	This review surveys robotics-driven advances in CNC machining. Automated material handling and tool changing to adaptive control, quality inspection, data analytics, and collaborative robots are evaluated for their impacts on efficiency, precision, and safety.	# of triples: 262 # of unique entities: 380 # of unique relations: 129

chosen due to their ease of access and relatively low computational requirements; here B represents the number of parameters in billions. By spanning 3B, 7B, and 8B parameter tiers, we can isolate how model capacity and design choices interact with our knowledge graph augmentation. Evaluating these models across a broad spectrum of parameter scales enables us to quantify how the integration of supplementary knowledge graph information influences overall performance. To establish an upper bound on attainable performance, we also benchmark against state-of-the-art closed sourced models including GPT-4o [33], its smaller GPT-4o-mini, Gemini 2.0 Flash [47], and Gemini 2.0 Flash-

Table 3
Sampling-based quantitative evaluation of triple extraction quality.

Metric	Estimate	95% CI	Support
Precision	0.845	[0.806, 0.877]	338/400
Recall	0.714	[0.640, 0.778]	115/161
F1	0.774	—	—

Lite [48]. These comprehensive models enables us to quantify precisely how supplemental structured knowledge closes the gap between open source baselines and leading proprietary offerings. For our semantic embedding model we chose multi-qa-MiniLM-L6-cos-v1 due to which is a finetuned model specific for question-answering semantic search based on the original MiniLM model developed by Microsoft [49]; we also chose this model specifically for its low memory requirement in deployment.

4.3. Question Category

Following knowledge graph construction, we devised two question formats namely, multiple choice and open ended to evaluate each model's capabilities. Multiple choice questions present a controlled benchmark for retrieval precision, enabling objective measurement and direct comparison across models, while open-ended questions simulate real-world case studies by challenging models to perform generative synthesis and coherently integrate the retrieved information into comprehensive answers. For each question format, two categories of questions were created namely: content specific and machining specific. Content specific questions refer to the information written in the documents that do not require quantitative analysis. Machining specific questions refer to questions that require quantitative precision and decision-making based on numerical parameters. An example of each question in multiple choice format is given below:

Content Specific Question

Which statement best describes a key distinction between 3-axis and 5-axis CNC milling machines?

A. 3-axis machines are used for metals, while 5-axis machines are used exclusively for plastics.
B. 5-axis machines include tilt and rotation of the workpiece or tool, in addition to X, Y, and Z motion.
C. 3-axis machines are larger and require more floorspace than 5-axis machines.
D. 5-axis machines do not allow any vertical movement while 3-axis machines do.

Machining Specific Question

What is the decimal equivalent of Drill Size 91 and what cutting speed (SFM) is recommended for drilling operations on aluminum?

A. 0.0087 inches and 250 SFM
B. 0.0083 inches and 200 SFM
C. 0.0083 inches and 300 SFM
D. 0.0095 inches and 150 SFM

GPT-4o was implemented to generate each question from the selected documents. From the 294 questions 65 content specific multiple choice, 72 machining specific multiple choice, 111 content specific open ended, and 46 machining specific open ended. Figure 4 showcases the topic distribution of the questions across 13 different categories for a total of 280; it should be noted that several multiple-choice questions were also converted to open-ended variants to test both formats and so there are 280 unique questions. As shown from the figure, the distribution of the questions are fairly uniform with eleven of the thirteen categories containing exactly 20 questions each ($\approx 7.1\%$) and two with slightly large groups 'Feeds, Speeds, Programming & Shop Practice' ($\approx 11.8\%$) and 'Energy Consumption & Green Manufacturing' ($\approx 9.6\%$) to reflect their prevalence for manufacturing scenarios. The near-uniform distribution of the questions and breath of the categories provides a diverse representative test set while minimizing topic over-representation and reducing the risk of sampling bias across the tested LLMs.

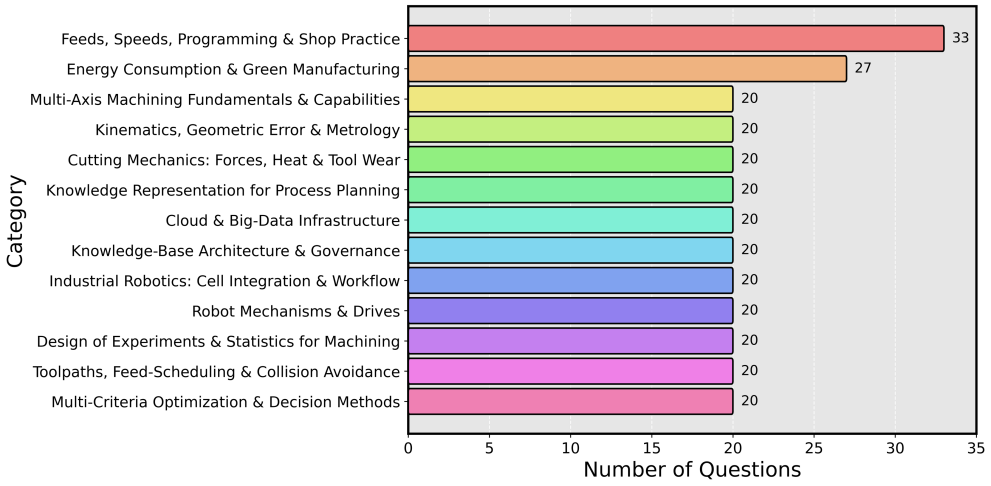

Figure 4: Topic distribution of the 280 unique industry-curated questions across 13 categories.

5. Experimental Results

ARKNESS is implemented using Python, PyTorch, and respective APIs for each of the chosen models using a Ryzen 9 7950X CPU and a Nvidia 4090 GPU. Figure 5 showcases the results between the use of a static graph with a top-K of only 10 and our dynamic knowledge graph retrieval algorithm for machining specific queries across the models for a) accuracy and b) F1-score metrics. As shown in the figure, across every model the dynamic retriever outperforms simply taking the top 10 triples, yielding consistent gains in both metrics. Rather than forcing a fixed amount of context for every question, the dynamic retriever adjusts the amount of required information to the query by expanding when additional dependencies are needed such as material, tool, and operation relationships and contracting when they are not. This avoids two common failure modes in machining guidance. First, it reduces the chance of missing a rare but decision critical constraint that would lead to an incorrect parameter recommendation. Second, it reduces the inclusion of irrelevant context that dilutes the signal and increases response time. The net effect is more consistent retrieval of the governing constraints that determine feasible feeds, speeds, and chip loads, which reduces back and forth during process planning, lowers the likelihood of off nominal setups, and reduces mid run parameter changes that can interrupt a job. Fewer clarifications and fewer exceptions also reduce release to floor holds, helping maintain start time reliability and keeping constrained machines producing saleable parts rather than waiting on parameter confirmation.

Table 4 provides a benchmark comparison between baseline model performance and results with the dynamic knowledge graph retrieval for both multiple choice datasets. As shown the knowledge graph produces the largest improvements on machining specific questions where the answer must respect hard constraints such as material properties, tool selection, and operation limits. This happens because retrieval aligns graph facts with the question entities and narrows the feasible machining window for feeds, speeds, chip load, and tolerance related constraints to spec consistent values. Constraining the feasible window reduces off nominal setups and reduces the chance that an operator selects a value that triggers chatter, poor surface finish, excessive tool wear, or a nonconformance rerun. Over repeated jobs, fewer parameter corrections reduce rework and inspection load, stabilize cycle time, and reduce the number of holds that interrupt work in process. For content specific questions, where baseline performance is already high, the knowledge graph primarily improves consistency and reduces rare planning errors that would otherwise require additional review cycles. Taken together, these improvements support more repeatable parameter selection and higher first pass success when translating planning guidance into production execution.

To translate these performance gains into operational terms, the improvement in machining-specific accuracy can be interpreted through production economics and cost-of-quality relationships. In machining economics, total machining time is commonly expressed as $T_m = T_c + T_{ct} + T_i$, where T_c is cutting time, T_{ct} is tool-changing time, and T_i is handling or idle time. Production cost is then obtained by multiplying these time elements by the relevant machine or labor cost rate. In addition, cost-of-poor-quality models define failure losses as the sum of internal and external failure costs, where internal failures include scrap, rework, and inspection before delivery, and external failures include

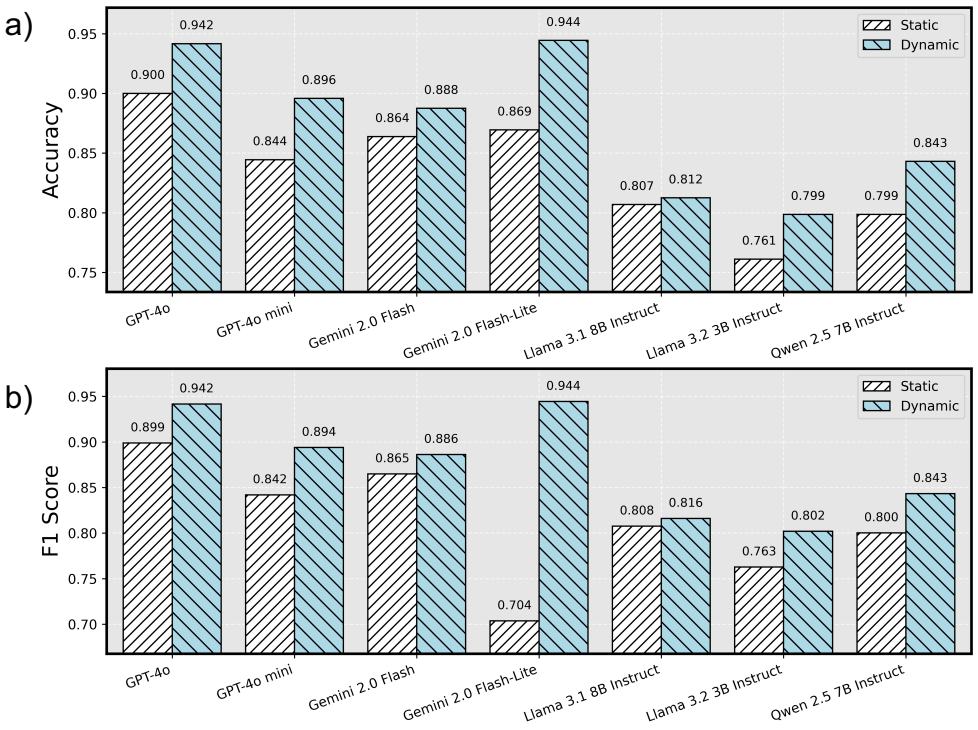

Figure 5: Results showcasing performance of use of dynamic knowledge graph compared to static graph for a) accuracy and b) F1-Score across machining specific questions.

warranty, returns, or customer-facing defects [50, 51, 52]. Using the largest observed machining-specific accuracy improvement in this study, $\Delta Acc = 0.166$, the expected number of avoided incorrect planning recommendations can be estimated as $\Delta E = N \times \Delta Acc$. For every $N = 1000$ process planning decisions, this corresponds to approximately 166 fewer incorrect recommendations. If each avoided error prevents only \$450 in combined scrap, rework, inspection, or lost machine time, the avoided quality loss cost is $S_q = \Delta E \times C_e = 166 \times 450 = \$74700$. Prior KG-based process reuse also reduced process-routing time from 50-80 minutes to approximately 15 minutes [27], which implies a planning-time saving of 35-65 minutes per plan. At a fully burdened planning labor rate of $C_l = \$60/\text{hr}$, where T_b denotes the baseline planning time before ARKNESS and T_a denotes the planning time after ARKNESS, the corresponding time saving can be estimated as $S_t = N \times ((T_b - T_a)/60) \times C_l$, yielding approximately \$35000-\$65000 per 1000 plans. Under this conservative scenario, the combined savings would be approximately \$109700-\$139700 per 1000 planning decisions before accounting for additional benefits from reduced tool wear, fewer delayed jobs, lower warranty exposure, or improved machine availability. Therefore, the quantitative gains reported in Table 4 are not only statistical improvements, but also indicate how KG-grounded LLMs can convert better parameter selection into measurable reductions in planning labor, nonproductive time, and quality related production losses.

Our next experiment examines the ability of LLMs to respond to open ended questions versus multiple choice questions. This evaluates performance when the model must generate a runnable answer rather than select among predefined options. Figure 6 provides the average results across 10 runs for each of the datasets for four different metrics. Semantic similarity evaluates closeness in meaning between the generated answer and the reference response by comparing their vector embeddings. ROUGE-1 measures unigram overlap, ROUGE-2 measures bigram overlap, and ROUGE-L measures the longest common subsequence between generated and reference text. The consistent improvement with knowledge graph augmentation indicates that grounding does not just improve the general idea of the response, but improves the completeness and specificity of what gets written. In practice, open ended answers that are missing a critical numeric detail or that present overly broad ranges force extra verification steps such as additional checks, trial cuts, or follow up clarification. By supplying the validated constraints and setpoints needed to answer, the knowledge graph reduces interpretation burden and reduces the frequency of mid run parameter changes. This keeps

Table 4
Accuracy and F1 for each model with and without the knowledge graph, with improvements over baseline shown in green parentheses.

Model	File	Baseline		Knowledge Graph	
		Accuracy	F1 Score	Accuracy	F1-Score
Gemini 2.0 Flash	Content Specific	0.938	0.940	0.958 (+0.020)	0.954 (+0.014)
	Machining Specific	0.805	0.806	0.889 (+0.084)	0.887 (+0.081)
Gemini 2.0 Flash-Lite	Content Specific	0.923	0.924	0.969 (+0.046)	0.969 (+0.045)
	Machining Specific	0.778	0.779	0.944 (+0.166)	0.944 (+0.165)
GPT-4o	Content Specific	0.938	0.939	0.954 (+0.016)	0.954 (+0.015)
	Machining Specific	0.833	0.834	0.944 (+0.111)	0.944 (+0.110)
GPT-4o mini	Content Specific	0.923	0.926	0.954 (+0.031)	0.953 (+0.027)
	Machining Specific	0.805	0.807	0.889 (+0.084)	0.886 (+0.079)
Llama 3.1 8B Instruct	Content Specific	0.877	0.877	0.969 (+0.092)	0.969 (+0.092)
	Machining Specific	0.658	0.668	0.711 (+0.053)	0.713 (+0.045)
Llama 3.2 3B Instruct	Content Specific	0.892	0.893	0.938 (+0.046)	0.938 (+0.045)
	Machining Specific	0.714	0.721	0.775 (+0.061)	0.780 (+0.059)
Qwen 2.5 7B Instruct	Content Specific	0.954	0.954	0.969 (+0.015)	0.970 (+0.016)
	Machining Specific	0.711	0.714	0.842 (+0.131)	0.845 (+0.131)

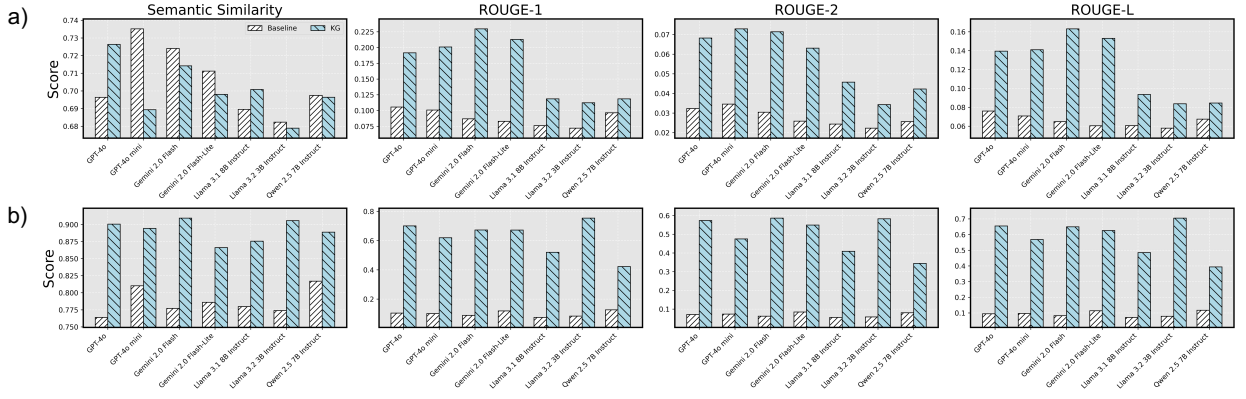

Figure 6: Comparison between baseline outputs with those augmented by knowledge graph data. a) for content specific questions, and b) for machining specific questions.

jobs moving with fewer holds and reduces the amount of nonproductive machine time spent on troubleshooting and retesting.

In addition to quantitative metrics such as semantic similarity and ROUGE, we include full answers for representative quantitative questions to show how each model actually uses retrieved graph knowledge. Table 5 and Table 6 show that baseline outputs often return wide parameter bands, which shift decision work back to the user and increase variability across operators and shifts. With knowledge graph augmentation, the models converge to specific values that match grounded references and can be executed directly. Moving from broad ranges to validated single value setpoints reduces setup time, reduces the chance of selecting an off nominal point that causes poor finish or premature tool wear, and reduces mid run adjustments that disrupt cycle time. Across repeated jobs, this raises repeatability, reduces inspection and rework load, and keeps constrained machines focused on producing saleable parts rather than correcting avoidable deviations.

Table 7 reports the computation profile for each model. The memory footprint is modest, with about 1.76 GB

Table 5
Open ended responses from open source models. Grounded quantitative answers are highlighted in yellow, baseline values in red, and knowledge graph augmented values in green.

Model / Question	For drilling operations on steel (4140), what cutting speed (SFM) is recommended?	For tapping operations on aluminum, what cutting speed (SFM) is recommended?	For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, what chip load is recommended?
Grounded Answer	For drilling operations on steel (4140), a cutting speed of 90 SFM is recommended.	For tapping operations on aluminum, a cutting speed of 100 SFM is recommended.	For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, a chip load of 0.004 inches per revolution is recommended.
Llama 3.1 8B Instruct	**Baseline:** The recommended cutting speed for drilling operations on 4140 steel ranges from 50 to 300 SFM, depending on the drill bit type and desired finish. **Knowledge Graph:** For drilling operations on steel (4140), the recommended cutting speed is 90 sfm.	**Baseline:** In general, the recommended cutting speed for tapping aluminum ranges from 50 to 150 SFM, depending on the specific conditions. **Knowledge Graph:** For tapping operations on aluminum, the recommended cutting speed is 100 sfm.	**Baseline:** A commonly cited range for chip loads on brass is between 0.002 and 0.008 inches per tooth (ipt). **Knowledge Graph:** For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, the recommended chip load is 0.004 ipr.
Llama 3.2 3B Instruct	**Baseline:** For drilling operations on 4140 steel, the recommended cutting speeds typically range from 60 to 120 SFM, with the optimal speed depending on the specific drilling conditions. **Knowledge Graph:** For drilling operations on steel (4140), a cutting speed of 90 SFM is recommended.	**Baseline:** A cutting speed of 30 to 50 FPM is commonly recommended for tapping operations on aluminum alloys, but the optimal cutting speed may vary depending on the application, tooling, and machining conditions. **Knowledge Graph:** For tapping operations on aluminum, a recommended cutting speed is 100 sfm.	**Baseline:** For milling operations on brass, a general rule of thumb is to use a chip load that is between 0.002 and 0.005 inches for a tool diameter between 0.5 and 1.0 inches. **Knowledge Graph:** Based on the provided context, for milling operations on brass with a tool diameter between 0.5 and 1.0 inches, the recommended chip load is 0.004 inches per revolution (ipr).
Qwen 2.5 7B Instruct	**Baseline:** The recommended SFM for drilling 4140 steel generally ranges between 25 and 35 feet per minute (FPM). **Knowledge Graph:** For drilling operations on steel (4140), a recommended cutting speed (SFM) is 90 sfm.	**Baseline:** For tapping operations on aluminum, a recommended cutting speed (Surface Footage per Minute, SFM) typically ranges between 100 to 200 SFM. **Knowledge Graph:** For tapping operations on aluminum, a recommended cutting speed (SFM) is 100 sfm.	**Baseline:** A common starting point for chip loads in this range is approximately 0.003 to 0.005 inches per tooth (IPM). **Knowledge Graph:** For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, a chip load of 0.004 inches per revolution (ipr) is recommended.

of system RAM primarily to hold the PostgreSQL knowledge graph, and an additional 0.604 GB of VRAM to host the embedding model and graph embeddings for closed source access. For local deployment, the open source models operate within a consumer GPU budget of under 10 GB VRAM, and end to end retrieval plus generation achieves a time to first token under 2.5 seconds. These characteristics support on prem inference with predictable per query cost, stable latency that keeps engineering and operator queues moving, and minimal incremental hardware outlay since a single consumer card can serve multiple concurrent requests with careful batching. Running locally also avoids cloud egress and API operating expenses, simplifies data governance for air gapped work, and reduces outage exposure tied to external services. Stable response time reduces waiting time for parameter checks and approvals and lowers the chance that jobs stall on information holds. Over sustained workloads, the modest hardware footprint reduces total cost of ownership and makes it practical to scale usage across cells without introducing a centralized bottleneck.

6. Conclusion

This paper introduces ARKNESS (Augmented Retrieval Knowledge Network Enhanced Search and Synthesis), a hybrid LLM-agnostic pipeline that couples an automatically constructed knowledge graph with a large language model to produce grounded and numerically precise answers for technical manufacturing queries. ARKNESS converts heterogeneous document formats into paragraph-level (subject, relation, object) triples while retaining full provenance and contextual information, and stores the resulting graph in a relational database to support efficient retrieval over

Table 6
Open ended responses from closed source models. Grounded quantitative answers are highlighted in yellow, baseline values in red, and knowledge graph augmented values in green.

Model / Question	For drilling operations on steel (4140), what cutting speed (SFM) is recommended?	For tapping operations on aluminum, what cutting speed (SFM) is recommended?	For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, what chip load is recommended?
Grounded Answer	For drilling operations on steel (4140), a cutting speed of 90 SFM is recommended.	For tapping operations on aluminum, a cutting speed of 100 SFM is recommended.	For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, a chip load of 0.004 inches per revolution is recommended.
Gemini 2.0-flash	**Baseline:** For 4140 steel, which is a medium-carbon, low-alloy steel known for its high strength and good hardenability, typical cutting speeds for drilling operations generally range from 70 to 120 SFM. **Knowledge Graph:** For drilling operations on steel (4140), a cutting speed of 90 SFM is recommended.	**Baseline:** For general-purpose tapping of aluminum alloys like 6061-T6, a common starting point is between 30 to 60 SFM. However, softer alloys like 1100 or 3003 can often be tapped at higher speeds, potentially up to 75 SFM or more. **Knowledge Graph:** For tapping operations on aluminum, a cutting speed of 100 SFM is recommended.	**Baseline**: The recommended chip load for milling brass with a tool diameter between 0.5 and 1.0 inches typically falls within the range of 0.001 to 0.004 inches per tooth. **Knowledge Graph:** For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, a chip load of 0.004 inches per revolution (ipr) is recommended.
Gemini 2.0-flash-lite	**Baseline:** For high-speed steel (HSS) drills, a starting point for the cutting speed when drilling 4140 steel is typically in the range of 40 to 60 surface feet per minute (SFM) for the steel in a hardened state. **Knowledge Graph:** For drilling operations on steel (4140), the recommended cutting speed is 90 SFM.	**Baseline:** For most aluminum alloys, a starting point for the cutting speed is typically in the range of 50 to 200 surface feet per minute (SFM). **Knowledge Graph:** For tapping operations on aluminum, a cutting speed of 100 SFM (Surface Feet per Minute) is recommended.	**Baseline:** For milling brass with a tool diameter between 0.5 and 1.0 inches, a chip load of 0.003 to 0.010 IPT is a good starting point. **Knowledge Graph:** For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, a chip load of 0.004 inches per revolution (ipr) is recommended.
GPT-4o	**Baseline:** For annealed AISI 4140 steel, which typically has a hardness of around 150-200 Brinell, the recommended cutting speed is usually in the range of 70 to 100 surface feet per minute (SFM) when using high-speed steel (HSS) drill bits. **Knowledge Graph:** For drilling operations on steel (4140), a cutting speed of 90 SFM is recommended.	**Baseline:** The recommended cutting speed for tapping aluminum typically ranges from 75 to 150 surface feet per minute (SFM). **Knowledge Graph:** For tapping operations on aluminum, a cutting speed of 100 surface feet per minute (SFM) is recommended.	**Baseline:** For general milling operations on brass with a tool diameter between 0.5 and 1.0 inches, a typical chip load recommendation would range from approximately 0.002 to 0.010 inches per tooth (IPT). **Knowledge Graph:** For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, the recommended chip load is 0.004 inches per revolution (ipr).
GPT-4o-mini	**Baseline:** Typically, for drilling operations on 4140 steel, a cutting speed in the range of 60 to 100 SFM is often recommended when using high-speed steel (HSS) drill bits. **Knowledge Graph:** For drilling operations on steel (4140), the recommended cutting speed is 90 surface feet per minute (SFM).	**Baseline:** For most aluminum alloys, a common recommendation for tapping speeds is in the range of 100 to 300 SFM. **Knowledge Graph:** For tapping operations on aluminum, the recommended cutting speed is 100 surface feet per minute (SFM).	Baseline: For brass, which is a relatively soft and ductile material, the recommended chip load generally falls within the range of 0.002 to 0.008 inches per tooth. **Knowledge Graph:** For milling operations on brass with a tool diameter between 0.5 and 1.0 inches, the recommended chip load is 0.004 inches per revolution (ipr).

both structured triples and their supporting text.

We show that combining provenance-aware KG retrieval with vector-embedding search and beam search improves the evidence available to the LLM at generation time. Across multiple-choice questions spanning content-specific and machining-specific queries, ARKNESS improves accuracy by up to 16.7% and F1-score by up to 16.5%. For open-ended questions, improvements of up to 8.9× in ROUGE-L indicate stronger alignment with reference answers, suggesting that structured retrieval helps the model produce responses that are both more faithful to source material and more technically consistent.

Beyond empirical gains, ARKNESS provides a practical mechanism for reducing hallucinations in settings where

Table 7
Computation requirements for each model.

Model	Max VRAM Required (GB)	Max RAM Required (GB)	Time to First Token (s)	Tokens Per Second (toks/s)
Gemini 2.0 Flash	0.604	1.76	0.881	144
Gemini 2.0 Flash-Lite	0.604	1.76	0.790	147
GPT-4o	0.604	1.76	1.21	69.8
GPT-4o mini	0.604	1.76	1.09	72.6
Llama 3.1 8B Instruct	8.57	1.76	2.21	8.78
Llama 3.2 3B Instruct	4.57	1.76	1.16	12.9
Qwen 2.5 7B Instruct	8.76	1.76	2.25	9.46

correctness and traceability matter. By retrieving compact, auditable evidence together with its provenance, the framework supports downstream applications such as process planning, troubleshooting, and decision support in CNC machining and production workflows, while enabling users to trace generated claims back to specific source contexts. This evidence-first design shifts the LLM's role from generating answers from latent knowledge to synthesizing responses from retrieved, verifiable technical material, which is particularly important when numerical values, process parameters, and operating constraints must be correct. In addition, the structured KG representation enables controllable retrieval and aggregation across documents, allowing ARKNESS to surface consistent parameter definitions, compare alternative recommendations, and expose potential conflicts in source material rather than masking them in fluent text. These properties make ARKNESS well-suited for human-in-the-loop settings, where engineers can quickly audit citations, validate assumptions, and update or override extracted facts, thereby improving both trust and usability in real manufacturing deployments.

This deployment value becomes especially important as the knowledge graph grows beyond the scale evaluated in this study. Industrial implementations may eventually include millions of relationships spanning machines, materials, cutting tools, inspection records, maintenance logs, sensor streams, and historical process plans. ARKNESS is designed to remain practical in such settings because it does not require full-graph traversal at inference time. Instead, the framework first narrows the candidate space using semantic similarity, then retrieves a compact evidence subgraph through adaptive graph traversal before passing only the most relevant context to the LLM. This design limits unnecessary retrieval while preserving important neighboring relationships that may affect the final recommendation. In larger deployments, the same architecture can be further supported through vector indexing, graph partitioning by material or process family, caching of frequently retrieved subgraphs, and incremental graph updates rather than full reconstruction. As a result, growth in graph size does not necessarily translate into proportional growth in inference cost. The framework can therefore scale as a searchable decision-support layer, where the LLM receives concise and auditable evidence rather than an unfiltered collection of all available manufacturing knowledge.

Although this study focuses on CNC machining, the underlying framework is not limited to subtractive manufacturing. The central workflow of ARKNESS consists of extracting domain knowledge from heterogeneous technical documents, representing that knowledge as contextualized and provenance-linked graph triples, retrieving the most relevant evidence for a user query, and using that evidence to ground LLM-based response generation. These steps are broadly applicable to other manufacturing domains where decisions depend on technical parameters, process constraints, material behavior, and traceable engineering knowledge. For example, in additive manufacturing, the graph could represent relationships among material properties, build parameters, defect mechanisms, and post-processing requirements; in welding, it could encode links among joint geometry, heat input, shielding gas, filler material, and quality outcomes; and in robotic assembly or maintenance planning, it could connect equipment states, task sequences, failure modes, and corrective actions. The specific entities and relationships would change across domains, but the core mechanism of combining structured retrieval with natural language reasoning would remain the same. This suggests that ARKNESS should be viewed as a transferable framework for explainable AI-assisted manufacturing decision support, provided that sufficient domain-specific documentation and validation procedures are available.

While the preceding results highlight the accuracy, deployment, and operational value of ARKNESS, several limitations remain that motivate future work. Although provenance-aware retrieval and conservative verification reduce errors, the constructed KG can still contain residual noise when evidence is implicit, spans multiple sentences, or is expressed through domain-specific paraphrases, and some verification thresholds and retrieval hyperparameters are currently selected empirically and may require re-tuning as the corpus or domain shifts. The current workflow is also predominantly text-centric and evaluated on English-language materials, and it does not yet represent temporal

or condition-dependent knowledge (e.g., time-varying telemetry or operating regimes), nor does it treat common non-text evidence such as technical drawings, CAD files, images, vibration or acoustic traces, and dense tabular datasheets as first-class, graph-aligned signals. Future work will expand ingestion and evaluation to multilingual corpora and queries, incorporate cross-lingual entity linking and language-agnostic identifiers, and extend the KG to multimodal and temporal evidence by lifting structured information from CAD and images, sensor streams, and tables or telemetry into graph-linked nodes and relations, supported by cross-modal embeddings for retrieval. We will accompany these extensions with broader benchmarks and provenance-aware evaluation protocols, enabling ARKNESS to scale to more diverse manufacturing settings while strengthening reliability, coverage, and traceability.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported under Cooperative Agreement W56HZV-21-2-0001 with the US Army DEVCOM Ground Vehicle Systems Center (GVSC), through the Virtual Prototyping of Autonomy Enabled Ground Systems (VIPR-GS) program, and by the National Science Foundation, United States (Grant No. 2434519).

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited. OPSEC10310

Disclaimer: Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA and shall not be used for advertising or product endorsement purposes.

References

[1] M. Wiessner, P. Blaser, S. Böhl, J. Mayr, W. Knapp, K. Wegener, Thermal test piece for 5-axis machine tools, Precision Engineering 52 (2018) 407–417.

[2] D. Hoang, H. Errahmouni, H. Chen, S. Rachuri, N. Mannan, R. ElKharboutly, M. Imani, R. Chen, F. Imani, Hierarchical representation and interpretable learning for accelerated quality monitoring in machining process, CIRP Journal of Manufacturing Science and Technology 50 (2024) 198–212.

[3] Z. Chen, D. Hoang, R. Chen, F. Imani, Distributed Hyperdimensional Computing for Real-Time Data Aggregation and Interpretable Quality Monitoring in Manufacturing, in: ASME International Mechanical Engineering Congress and Exposition, vol. 88605, American Society of Mechanical Engineers, V002T03A092, 2024.

[4] Z. Li, Y. Dai, Z. Sun, C. L. Guan, T. Lai, H. Xu, X. Zhou, A sub-micron precision machining and measurement method of long travel metal guideways, Journal of Manufacturing Processes 133 (2025) 947–956.

[5] ForInsights Consultancy, Computer Numerical Control (CNC) Machine Market Forecast 2030, https://www.forinsightsconsultancy.com/reports/computer-numerical-control-cnc-machine-market/, accessed: 2025-05-12, 2023.

[6] D. Hoang, N. Mannan, R. ElKharboutly, R. Chen, F. Imani, Edge cognitive data fusion: From in-situ sensing to quality characterization in hybrid manufacturing process, in: International Manufacturing Science and Engineering Conference, vol. 87240, American Society of Mechanical Engineers, V002T06A029, 2023.

[7] D. Hoang, H. Chen, M. Imani, R. Chen, F. Imani, Brief Paper: Multi-Task Brain-Inspired Learning for Interlinking Machining Dynamics With Parts Geometrical Deviations, in: International Manufacturing Science and Engineering Conference, vol. 88117, American Society of Mechanical Engineers, V002T05A012, 2024.

[8] K. Spanaki, D. Dennehy, T. Papadopoulos, R. Dubey, Data-driven digital transformation in operations and supply chain management, International Journal of Production Economics 284 (2025) 109599.

[9] Siemens AG, The True Cost of Downtime 2022, Tech. Rep., Siemens, URL https://assets.new.siemens.com/dics-b10153-00-7600truecostofdowntime2022-144.pdf, 2022.

[10] N. C. Nwasuka, U. Nwaiwu, Computer-based production planning, scheduling and control: a review, Journal of Engineering Research 12 (1) (2024) 275–280.

[11] M. Raza, Z. Jahangir, M. B. Riaz, M. J. Saeed, M. A. Sattar, Industrial applications of large language models, Scientific Reports 15 (1) (2025) 13755.

[12] S. Fosso Wamba, C. Guthrie, M. M. Queiroz, S. Minner, ChatGPT and generative artificial intelligence: an exploratory study of key benefits and challenges in operations and supply chain management, International Journal of Production Research 62 (16) (2024) 5676–5696.

[13] B. Q. Tan, K. Kang, R. Y. Zhong, When to use large language models for digital manufacturing in supply chains?, International Journal of Production Economics (2026) 109932.

[14] Z. Jin, Z. Zhou, H. Wu, Enhancing digital manufacturing efficiency and dominance relation driven big Data analytics, International Journal of Production Economics (2025) 109790.
[15] J. Feng, Y. Ning, Z. Wang, G. Li, S. X. Xu, ChatGPT-enabled two-stage auctions for electric vehicle battery recycling, Transportation Research Part E: Logistics and Transportation Review 183 (2024) 103453.
[16] M. Abbaszadeh Nakhost, A. Bloemer, S. Minner, Full automation or just a copilot? A study of large language models on spare parts demand forecasting with OpenAI o1-mini, International Journal of Production Research (2025) 1–16.
[17] K. Šket, D. Potočnik, M. Ficko, S. Klančnik, Enhancing G-Code Programming in Cnc Machining Using Chatgpt: A Comparative Study of Gpt-3.5 and Gpt-4.0, Available at SSRN 4940034 .
[18] J. Jeon, Y. Sim, H. Lee, C. Han, D. Yun, E. Kim, S. L. Nagendra, M. B. Jun, Y. Kim, S. W. Lee, et al., ChatCNC: Conversational machine monitoring via large language model and real-time data retrieval augmented generation, Journal of Manufacturing Systems 79 (2025) 504–514.
[19] R. Rosati, F. Antonini, N. Muralikrishna, F. Tonetto, A. Mancini, Improving industrial question answering chatbots with domain-specific llms fine-tuning, in: 2024 20th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), IEEE, 1–7, 2024.
[20] S. Kanimozhi, Y. Sriker, et al., Explorative Deployment of Fine-Tuned Large Language Model for On-Site Computerized Numeric Control Machine Operator Assistance, in: 2024 IEEE Silchar Subsection Conference (SILCON 2024), IEEE, 1–6, 2024.
[21] P. T. Amarasinghe, S. Nguyen, Y. Sun, S. S. Arisian, D. Alahakoon, Business optimization for digital manufacturing: A fine-tuned large language model approach, International Journal of Production Economics (2026) 109934.
[22] R. Dou, G. Nan, Y. Pan, X. Liu, Balancing privacy considerations and customization preferences for consumer: LLMs adoption and coordination strategies in supply chains, International Journal of Production Economics (2025) 109827.
[23] Y. Xiao, S. Zheng, J. Shi, X. Du, J. Hong, Knowledge graph-based manufacturing process planning: A state-of-the-art review, Journal of Manufacturing Systems 70 (2023) 417–435.
[24] P. Wen, Y. Ma, R. Wang, Systematic knowledge modeling and extraction methods for manufacturing process planning based on knowledge graph, Advanced Engineering Informatics 58 (2023) 102172.
[25] D. Hoang, D. Gorsich, M. Castanier, F. Imani, Vector-Symbolic Knowledge Graphs for Enhanced Memorization and Reasoning in Digital Manufacturing, Available at SSRN 5097516 .
[26] L. Wang, H. Cheng, R. Wang, X. Huang, Machining Scheme Selection of Features Based on Process Knowledge Graph and Improved Cosine Similarity Matching, Machines 13 (3) (2025) 188.
[27] J. Guo, J. Wu, J. Bian, Q. He, Knowledge Graph-Based Machining Process Route Generation Method, in: International Conference on Human-Computer Interaction, Springer, 35–48, 2023.
[28] C. Cai, Z. Jiang, H. Wu, J. Wang, J. Liu, L. Song, Research on knowledge graph-driven equipment fault diagnosis method for intelligent manufacturing, The International Journal of Advanced Manufacturing Technology 130 (9) (2024) 4649–4662.
[29] Y. Li, H. Zhao, H. Jiang, Y. Pan, Z. Liu, Z. Wu, P. Shu, J. Tian, T. Yang, S. Xu, et al., Large language models for manufacturing, arXiv preprint arXiv:2410.21418 .
[30] P. Wang, J. Karigiannis, R. X. Gao, Ontology-integrated tuning of large language model for intelligent maintenance, CIRP annals 73 (1) (2024) 361–364.
[31] C. Qiu, B. Li, H. Liu, S. He, C. Hao, A novel method for machine tool structure condition monitoring based on knowledge graph, The International Journal of Advanced Manufacturing Technology 120 (1) (2022) 563–582.
[32] C. Auer, M. Lysak, A. Nassar, M. Dolfi, N. Livathinos, P. Vagenas, C. B. Ramis, M. Omenetti, F. Lindlbauer, K. Dinkla, et al., Docling Technical Report, arXiv preprint arXiv:2408.09869 .
[33] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al., Gpt-4o system card, arXiv preprint arXiv:2410.21276 .
[34] M. Soori, F. K. G. Jough, R. Dastres, B. Arezoo, A Review in Capabilities and Challenges of 5-Axis CNC Milling Machine Tool Operations, Preprint .
[35] Y. Ye, T. Hu, C. Zhang, W. Luo, Design and development of a CNC machining process knowledge base using cloud technology, The International Journal of Advanced Manufacturing Technology 94 (2018) 3413–3425.
[36] C. Guo, Exploring the Application of Industrial Robots in CNC Machining, Journal of Global Humanities and Social Sciences 4 (5) (2023) 231–235, doi:\let\@tempa\bibinfo@X@doi10.61360/BoniGHSS232014990503, URL http://ojs.bonfuturepress.com/index.php/GHSS/article/view/1499.
[37] Autodesk, Inc., Fundamentals of CNC Machining: A Practical Guide for Beginners, United States, URL https://academy.titansofcnc.com/files/Fundamentals_of_CNC_Machining.pdf, desk Copy. Document Number: 060711, 2014.
[38] I. Nugrahanto, H. Gunawan, H.-Y. Chen, Innovative approaches to sustainable computer numeric control machining: a machine learning perspective on energy efficiency, Sustainability 16 (9) (2024) 3569.
[39] M. Ihsan, Y. Sumantri, Y. Irawan, Integration of Taguchi and Promethee for CNC Milling Machining Parameter Optimization on AA6061, International Journal of Mechanical Engineering Technologies and Applications 5 (1) (2024) 96–107.
[40] S. Zhang, J. Bai, Research on CNC programming and machining process based on CAD/CAM technology, Applied Mathematics and Nonlinear Sciences 9 (1) (2024) 1–18, doi:\let\@tempa\bibinfo@X@doi10.2478/amns-2024-0516, URL https://doi.org/10.2478/amns-2024-0516.
[41] H. Wu, X. Wang, X. Deng, H. Shen, X. Yao, Review on design research in CNC machine tools based on energy consumption, Sustainability 16 (2) (2024) 847.
[42] M. Soori, F. K. G. Jough, R. Dastres, B. Arezoo, Robotical automation in CNC machine tools: a review, acta mechanica et automatica 18 (3).
[43] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 .
[44] Meta AI, Llama 3.2 Model Card, https://github.com/meta-llama/llama-models/tree/main/models/llama3_2, version 3.2, re-

leased 25 Sept 2024, 2024.
[45] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, arXiv preprint arXiv:2407.21783 .
[46] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al., Qwen2. 5 technical report, arXiv preprint arXiv:2412.15115 .
[47] Google DeepMind, Gemini 2.0 Flash, URL https://deepmind.google/technologies/gemini/flash/, 2025.
[48] Google DeepMind, Gemini Flash Lite, URL https://deepmind.google/technologies/gemini/flash-lite/, 2025.
[49] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, Advances in neural information processing systems 33 (2020) 5776–5788.
[50] W. A. Knight, G. Boothroyd, Fundamentals of metal machining and machine tools, CRC Press, 2019.
[51] A. Bhattacharyya, Metal Cutting: Theory and Practice, New Central Book Agency, ISBN 9788173812002, 2017.
[52] A. Chattopadhyay, Machining and machine tools (With CD), John Wiley & Sons, 2011.