

Article

STAR: Steelmaking Task-Aware Routing for Multi-Agent LLM Expert Systems

Wenyuan Liu ¹, Chengyan Huang ¹, Songlei Wang ¹, Lin Wang ^{1,*}, Fanjie Meng ², Minghui Li ², Haoning Zhang ² and Qiang Zheng ²

¹ School of Artificial Intelligence, Yanshan University, 438 Hebei Street, Qinhuangdao 066004, China; wylu@ysu.edu.cn (W.L.); huangchengyan@stumail.ysu.edu.cn (C.H.); wsl@stumail.ysu.edu.cn (S.W.)

² State Key Laboratory of Metallurgical Intelligent Manufacturing System, 72 West 4th Ring Road South, Beijing 100071, China; mmmffjjj@126.com (F.M.); lft_happiness@163.com (M.L.); zzzzhnnn@163.com (H.Z.); zqawszqaws@126.com (Q.Z.)

* Correspondence: wlin@ysu.edu.cn

Abstract

Steelmaking involves long, tightly coupled process chains and specialized domain knowledge, making it difficult in practice for a single general-purpose LLM to consistently align engineers' queries with the correct process stage. This paper presents STAR, an industry-oriented multi-stage process-domain router for steel metallurgy, and provides an integration blueprint that maps routing labels to domain-specific prompting and retrieval scopes in a router-plus-agents architecture. We construct a quality-controlled metallurgical corpus from textbooks, manuals, and papers via OCR and multi-dimensional text-quality scoring. Based on this corpus, we build an LLM-assisted pipeline to synthesize query–domain pairs for eight fine-grained process domains under domain definitions/keywords and format constraints, and index all queries in a shared embedding space with FAISS. We design a three-stage router: (1) a lightweight filter using chat rules and a nearest-neighbor distance threshold to separate steel-related queries from general ones, (2) a kNN label-voting router whose confidence is derived from the Top-*k* neighbor label concentration, and (3) an LLM-based refinement step for low-confidence cases with safe fallback. Experiments on 3136 steel-domain queries and approximately 2000 general queries show that STAR achieves 0.921 Top-1 accuracy and 0.899 macro-F1 on 8-way fine-grained steel-domain routing, and achieves a steel-query recall of 0.999 for steel-versus-general filtering (queries routed to `general_llm` in deployment). In this work, we primarily evaluate routing quality and efficiency; end-to-end answer quality evaluation of downstream agents is left for future work.

Keywords: steelmaking; large language models; process-domain routing; retrieval-augmented generation; industrial question answering



Academic Editor: Alberto Fernandez Hilario

Received: 9 January 2026

Revised: 4 February 2026

Accepted: 6 February 2026

Published: 7 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The iron and steel industry is a representative process industry characterized by long process chains, complex stages, and strongly coupled operating parameters. From raw material preparation to blast-furnace ironmaking, steelmaking and secondary refining, continuous casting, and rolling for heat treatment and quality control, each stage involves substantial domain knowledge and many practice-driven rules [1]. For decades, when frontline operators and process engineers encountered production issues, they largely relied

on printed textbooks, technical papers, a limited number of internal procedures, and personal experience to make decisions. This “people searching for knowledge” paradigm is time-consuming and labor-intensive, and it depends heavily on senior experts’ availability, making it difficult to meet modern steel enterprises’ demands for timely, stable, and reproducible decision-support [2].

In recent years, large language models (LLMs) have shown strong capabilities in open-domain dialogue, code generation, and knowledge-intensive question answering, providing new building blocks for intelligent question answering and decision-support systems in the steel industry [3,4]. A common practice is to combine a general-purpose LLM with retrieval-augmented generation (RAG), allowing the model to consult an external knowledge base during answer generation and partially mitigating limited parametric knowledge [5]. However, when general-purpose LLMs and generic RAG pipelines are applied to steelmaking and metallurgical scenarios, several challenges remain. First, *domain expertise is uneven*: under limited metallurgy-domain coverage, the model may produce plausible but incorrect statements, which is undesirable in high-stakes production settings. Second, *process-stage granularity is often too coarse*: the steelmaking chain spans many sub-domains, and a general-purpose model may fail to distinguish whether a query concerns, for example, blast-furnace operation, continuous casting defects, or controlled rolling and cooling, leading to mismatches between retrieved knowledge and the intended process stage. Third, *controllability and scalability are limited*: under a “single-model + single-retrieval” design, there is no explicit mechanism to route queries to stage-specific responsibilities, which hinders interpretability and the integration of specialized components for different process stages [3,5,6].

To address these issues, we take a routing perspective and study how to map a user’s natural-language query to an appropriate process domain in the iron and steel metallurgy workflow. Rather than relying on a monolithic model to handle all queries, we decompose the steelmaking process into several fine-grained professional domains and design a practical routing module that dispatches queries to the corresponding domain label. These labels can be used to select domain-scoped retrieval indexes and domain-specific system prompts, forming a router-plus-agents *integration blueprint* that improves process-stage alignment and controllability in an industrial setting [5,7,8]. In this paper, our primary focus is the router itself, including its data construction, decision mechanism, and routing performance, while the downstream agent behaviors are treated as pluggable components that can be iterated independently.

A key practical challenge in this domain is that a large fraction of metallurgy textbooks and technical documents are available only as scanned copies, often with irregular layouts, mixed text and figures, and widely varying quality. This complicates systematic corpus construction and downstream retrieval/routing [9,10]. Motivated by industrial literature and application scenarios, we build a metallurgy corpus with OCR-based preprocessing and text-quality scoring, define a fine-grained domain labeling scheme along the process chain, and propose a three-stage routing framework for steel-related queries. Specifically, the router first distinguishes steel-specific technical queries from general chit-chat with a lightweight filter; it then uses vector retrieval to estimate candidate domains; finally, it invokes an LLM-based refinement step for low-confidence or ambiguous cases [5]. This design combines statistical evidence from a domain-labeled vector space with semantic signals from pretrained models [3,5].

Building on this route, we implement a steel-domain router that predicts the process domain implied by a user’s query and outputs a routing label that can be used to select domain-scoped retrieval and prompting configurations. In this paper, we quantitatively evaluate routing quality and efficiency; end-to-end answer quality evaluation of

downstream agents is beyond the current scope and is left for future work. The main contributions of this work are summarized as follows:

- We present an end-to-end workflow for building a steelmaking-oriented routing framework, covering OCR-based text preprocessing and quality scoring, fine-grained process-domain definitions, LLM-assisted query construction, and domain-labeled vector index construction, culminating in a practical multi-stage router design [9,10].
- We construct a fine-grained steel-domain question set and a domain-labeled vectorized knowledge space. By organizing metallurgy texts alongside process domains, we generate and label typical engineering queries and build a FAISS-based index with domain metadata, providing a data foundation for routing evaluation and domain-scoped retrieval integration [3,6].
- We design a three-stage process-domain router that combines rule-based heuristics, retrieval-based neighbor voting, and LLM-based refinement. We further provide a router-plus-agents integration blueprint in which routing labels map-to-domain-specific prompting and retrieval scopes, enabling stage-aware query dispatching while remaining extensible to additional domains and components [5,7,8].

2. Related Work

In recent years, the application of large language models (LLMs) to intelligent manufacturing and process industries has attracted increasing attention. Li et al. review the potential and application pathways of LLMs in manufacturing, noting their promise for knowledge retrieval and decision-support across design, production, and service, while industrial deployment remains at an early stage [11]. Zhang et al. survey LLM-enabled next-generation intelligent manufacturing from the perspective of “potential–path–challenges,” highlighting data quality, security, and reliability as prerequisites for large-scale adoption [12,13]. At the system level, many efforts follow a “single model + RAG/knowledge graph + LLM interface” paradigm. For example, LLM-MANUF fine tunes and ensembles multiple small-scale LLMs on domain data to support decision-making in manufacturing operations and maintenance [14]. In metal additive manufacturing and broader mechanical engineering, systems such as the LLaMA2-7B-based AM question answering system AMGPT, knowledge graph-based decision-support and knowledge-service systems [15–17], and multimodal approaches such as MechRAG have been proposed [18]. These studies indicate that LLMs are becoming an important foundation for industrial knowledge services. However, many existing works still treat complex industrial workflows as a monolithic whole, with relatively limited emphasis on fine-grained process-domain partitioning and domain-labeled question resources along the process chain. In particular, building multi-domain question sets and labeled vector spaces that cover the full iron- and steelmaking workflow is less explored.

Research specifically targeting iron and steel metallurgy has also begun to emerge. Fu et al. integrate vision–language models with LLMs to build a smelting process management system that combines defect detection, data analysis, and intelligent question answering, enabling linkage between molten-steel defect images and process-data analysis [19]. Other studies discuss technical roadmaps for “LLMs empowering the steel industry” and propose evaluation considerations for large models in the steel sector [13]. Benchmarks such as StiBench assess both general-purpose and industry models in terms of conceptual understanding, mechanistic reasoning, and process-related question answering, and report that general-purpose LLMs still exhibit gaps in expertise and reliability for steel-domain tasks [20]. Overall, these works support the need for specialized LLM-based decision-support in the steel domain. Meanwhile, at the system level, many approaches adopt either a stage-specific expert system or a “single model + retrieval” architecture for

steel Q&A. Fewer works explicitly decompose the long-process steelmaking workflow into multiple process-domain agents that can evolve independently, or introduce an explicit routing module that dispatches queries among such agents. In contrast, starting from OCR-extracted textbook and paper texts, this paper performs fine-grained domain partitioning along the process chain, constructs a domain-labeled query set and a vector space, and designs a multi-stage router that dispatches queries to the process-domain agent responsible for the corresponding stage.

From a methodological perspective, our routing perspective relates to recent work on LLM routing and test-time scaling. RouterBench, RouteLLM, UniRoute, and R2R study multi-model routing evaluation, routing between stronger and weaker models using preference data, routing over dynamic model pools, and token-level mixed inference with large and small models, respectively, exploring efficiency–accuracy trade-offs across general-purpose LLMs [21–24]. In contrast, our focus is process-domain routing in a process-industry setting, where the “experts” correspond to heterogeneous process stages with distinct procedures and knowledge bases. The routing objective is therefore not only computational efficiency, but also process-stage alignment and controllability in a stage-aware agent architecture. At the system level, our “router + multi-agent experts” architecture is related to multi-agent and industrial-agent frameworks such as AutoGen, ProcessCarbonAgent, ChatTwin, and DeFACT [25–28]. Unlike systems that mainly rely on manually designed workflows, we leverage a domain-labeled query set and statistical signals from a labeled vector space to build a three-stage router—lightweight filtering, retrieval-based neighbor voting, and LLM-based refinement for ambiguous cases—and connect this router to a process-domain agent architecture. This design targets the structure of the iron- and steelmaking workflow and provides a practical pathway for building stage-aware routing and decision-support systems in process-industry scenarios.

3. Method

This section details the construction of the proposed steel-domain process-router, including data preprocessing, domain-specific question construction, the vectorized knowledge space, and the multi-stage routing mechanism, as well as an integration blueprint that connects routing labels to domain-scoped retrieval and prompting configurations. The overall workflow follows the technical route outlined in Section 1 and is consistent with the vector-retrieval and routing logic used in our engineering implementation. Specifically, the system first extracts domain knowledge from steel metallurgy textbooks and the technical literature and constructs a structured corpus through OCR preprocessing, domain partitioning, and question generation [1,9,10]. It then builds a domain-labeled vectorized knowledge space using a Chinese sentence-embedding model and FAISS [29,30]. Finally, the multi-stage router maps user queries to process-domain labels to support stage-aware dispatching in steel-domain LLM systems [5,7,8].

3.1. Overall System Architecture

The system adopts a hierarchical “router + domain expert agents + general-purpose agent” architecture, tightly coupled with metallurgy corpus construction and the vectorized knowledge space [6–8]. The end-to-end workflow is illustrated in Figure 1. After the front-end receives a user’s natural-language query, it is first fed into the steel-domain intelligent router, which determines whether the query falls within the professional iron and steel metallurgy scenario. If it does, the router further predicts a fine-grained process-domain label, such as raw materials and ironmaking, steelmaking and secondary refining, continuous casting and slab quality, rolling and controlled rolling/cooling, heat treatment and microstructure–property relationships, steel grade design and composition control, defect

analysis and quality control, production organization, and green/low-carbon metallurgy. Based on the routing result, the query is forwarded to the corresponding domain expert agent; if the query is judged to be non-steel-related or casual chit-chat, it is instead routed to the general-purpose agent [7,8]. Accordingly, the deployed router outputs 9 routing labels in total: 8 fine-grained steel process domains plus general_llm for non-steel queries.

In the engineering implementation, all domain expert agents and the general-purpose agent share the same underlying LLM instance, while their roles are differentiated through distinct system prompts and retrieval scopes [5,6]. The router is responsible only for outputting routing labels and confidence scores, whereas answer generation is carried out by downstream agents in combination with their respective domain knowledge bases [5]. This design provides a clear division of responsibilities among agents while maintaining system scalability without increasing the structural complexity of the underlying model [6–8].

In this work, we treat downstream agents as pluggable components whose behaviors depend on system prompts and retrieval scopes. We focus on evaluating the routing module, including steel versus general filtering and 8-way process-domain routing, because reliable stage alignment is a prerequisite for building robust agent-based industrial QA systems. End-to-end answer quality evaluation under different agent configurations is left for future work.

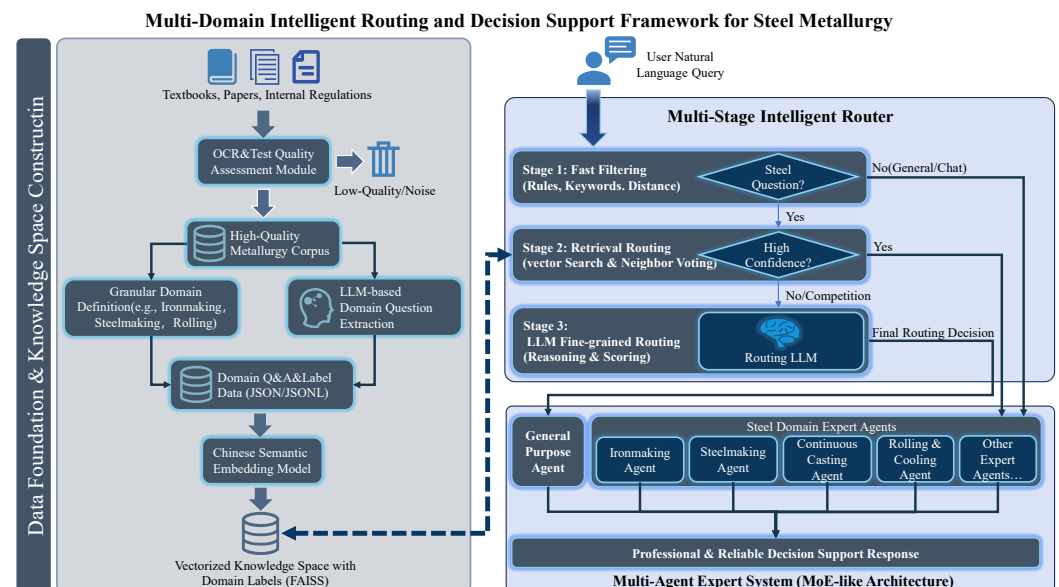


Figure 1. Overall architecture of the proposed steel-domain multi-stage router and a router-plus-agents integration blueprint. Left: pipeline for constructing the domain-labeled vector knowledge space, from OCR/quality assessment and granular domain definition to LLM-based question extraction and FAISS embeddings. Right: three-stage router (fast filtering, retrieval routing, LLM fine-grained routing) that outputs routing labels for selecting a general-purpose configuration or domain-scoped retrieval/prompting configurations.

3.2. Metallurgical Corpus Construction and Question Generation

3.2.1. OCR Text Preprocessing and Quality Assessment

To obtain a high-quality textual corpus covering the entire steelmaking process, we use scanned steel metallurgy textbooks, monographs, and academic papers as primary data sources and apply OCR tools to convert them into Markdown text [1,9,10]. Because scan quality varies substantially, directly using raw OCR outputs often introduces garbled characters, fragmented table residues, and truncated paragraphs; therefore, automatic quality control is necessary during corpus construction [9,10]. Our preprocessing pipeline includes removing headers, footers, and page numbers; performing coarse-grained filtering

of regions that are clearly tables; discarding abnormally short lines and lines consisting purely of digits; and recombining/splitting text at the paragraph level to rejoin sentences that were incorrectly broken across lines. These steps ensure that subsequent feature computation is performed on text segments that are semantically more complete [9,10].

Let an OCR text segment be denoted as x . We construct a multi-dimensional text-quality feature vector:

$$f(x) = (f_{\text{len}}(x), f_{\text{punc}}(x), f_{\text{conn}}(x), f_{\text{rep}}(x), f_{\text{zh}}(x), f_{\text{noise}}(x)), \quad (1)$$

where f_{len} measures the deviation of segment length from the document-level average, f_{punc} denotes the density of valid punctuation marks, f_{conn} characterizes the density of Chinese discourse connectives (e.g., 因此, 同时, 另一方面, 此外, 然而, 综上所述), f_{rep} measures the repetition of high-frequency words, f_{zh} is the proportion of Chinese characters, and f_{noise} measures abnormal characters (e.g., garbled symbols and consecutive punctuation marks). After normalizing each feature, we compute an overall quality score via linear weighting:

$$s(x) = \sum_j w_j f_j(x), \quad s(x) \in [0, 1], \quad (2)$$

where weights w_j are determined by combining small-scale manual sampling with heuristic tuning. Specifically, we select representative pages and ask annotators to label OCR outputs into categories such as “usable,” “needs revision,” and “unusable.” We then infer the relative importance of different features on this small labeled set and adjust w_j to penalize garbled segments and table fragments more strongly.

3.2.2. Corpus Statistics, Quality Evidence, and Compliance

To make the OCR corpus contribution measurable and reproducible, we report corpus-scale statistics, filtering effects, and basic compliance considerations.

Our raw OCR sources consist of scanned metallurgy textbooks/monographs and technical papers that cover the full steelmaking workflow, including upstream raw materials and ironmaking, steelmaking/refining, continuous casting, rolling, heat treatment, defect analysis, production organization, and green/low-carbon topics. In total, the raw OCR pipeline processes 36 documents with 7800 pages, producing 305,000 text segments before quality filtering.

We quantify the impact of the quality score $s(x)$ (Equation (2)) by reporting retention rates and a small human-evaluated quality sample. Specifically, we uniformly sample 500 OCR segments before filtering and 500 segments after filtering, and ask annotators to label each segment into three categories: *usable*, *needs revision*, and *unusable*. Table 1 summarizes the corpus scale, retention rate, and the proportion of usable text before/after filtering. This evidence supports that the proposed feature-based scoring reduces garbled text and table fragments while preserving professional content.

We observe three common OCR error modes: (i) *broken sentences* due to incorrect line breaks, (ii) *table residues* that appear as fragmented numbers and symbols, and (iii) *garbled characters* from low-resolution scans. These errors can distort semantic embeddings and thus harm both retrieval and routing. Our preprocessing and quality filtering primarily target (i)–(ii), and the abnormal-character feature $f_{\text{noise}}(x)$ penalizes (iii). In deployment, we log low-quality segments and update filtering heuristics iteratively.

The constructed OCR corpus is used for internal research and system development. We do not redistribute copyrighted full texts; only derived artifacts (e.g., embeddings and short text snippets necessary for retrieval/routing) are stored in the system. For any internal manuals or proprietary materials, we ensure usage permission and access control.

When releasing any benchmark data, we only share *expert-validated* questions and metadata that do not contain copyrighted passages, and provide the full-text corpus only upon permission and under the appropriate agreements.

Table 1. OCR corpus statistics and quality evidence.

Item	Description	Before Filtering	After Filtering
Documents	textbooks/monographs/papers	36	36
Pages	scanned pages processed	7800	7800
Segments	paragraph-level OCR segments	305,000	228,000
Retention rate	kept/total	—	0.75
Avg. segment length	characters per segment	210	240
Usable rate (human)	% labeled as usable	0.62	0.86
Unusable rate (human)	% labeled as unusable	0.21	0.05

If $s(x)$ falls below a predefined threshold τ_{ocr} , the segment is treated as low-quality and filtered out. We choose τ_{ocr} via grid search and manual spot checking, such that the retained text contains minimal obvious garbling while preserving as much professional content as possible. This process removes pages with severe noise, table fragments, and heavily truncated paragraphs, yielding a high-reliability metallurgy corpus $\mathcal{D}$. Subsequent domain question generation and vectorized knowledge-base construction are performed only on $\mathcal{D}$, improving robustness at the source.

3.2.3. Fine-Grained Domain Partitioning and Corpus Annotation

Based on the corpus $\mathcal{D}$, we partition the knowledge space into a set of fine-grained domains following the process structure of the iron and steel industry. Let the domain set be

$$\mathcal{C} = \{c^{(1)}, c^{(2)}, \dots, c^{(8)}\}. \quad (3)$$

Each domain corresponds to a specific stage in the steelmaking process, including raw materials and ironmaking, steelmaking and secondary refining, continuous casting and slab quality, rolling and controlled rolling/cooling, heat treatment, steel grade design and composition control, defect analysis and quality control, and production organization and green/low-carbon metallurgy. In implementation, each domain is described by a structured configuration (domain identifier, Chinese description, and typical keywords) that is kept consistent with the domain configuration used in the code, so that domain identifiers can be mapped to concrete process modules during routing.

For each metallurgy text segment $x \in \mathcal{D}$, we design prompts that guide the LLM to generate several practically relevant question formulations based on the context and to assign domain labels according to predefined domain descriptions and keyword lists, thereby constructing a domain-specific question dataset:

$$\mathcal{S} = \{(q_i, c_i)\}_{i=1}^N, \quad c_i \in \mathcal{C}, \quad (4)$$

where q_i is a natural-language question and c_i is its fine-grained domain label. To improve label consistency and interpretability, we provide domain descriptions and representative keywords during generation, and require the model to compare domain definitions before making a decision, thereby reducing randomness for boundary cases. The resulting dataset is stored in JSON/JSONL format, with domain labels aligned one-to-one with the metadata fields used in the vector-retrieval module. This supplies the router with statistical evidence in the form of historical neighbor-domain distributions and lays the groundwork for potential future extensions to supervised routing models.

3.3. Vectorized Knowledge Space and Retrieval Module

3.3.1. Semantic Encoding and Vector Space Construction

To exploit semantic similarity between metallurgical questions during routing, we use a Chinese sentence-embedding model (e.g., bge-large-zh) to encode domain-specific questions into vectors. Let the encoding function be $f_\theta(\cdot)$

$$\mathbf{h} = f_\theta(q) \in \mathbb{R}^d, \quad (5)$$

where f_θ is the pretrained sentence-embedding model and d is the embedding dimension. In our implementation, all vectors are ℓ_2 -normalized (i.e., $\|\mathbf{h}\|_2 = 1$), so that ℓ_2 distance and cosine similarity induce equivalent nearest-neighbor rankings. Normalization is performed once offline during index construction and persisted together with the metadata to avoid redundant computation during online inference.

For each question $q_i \in \mathcal{S}$, we compute $\mathbf{h}_i = f_\theta(q_i)$ and construct a domain-labeled vectorized knowledge space:

$$\mathcal{V} = \{(\mathbf{h}_i, \mathbf{m}_i)\}_{i=1}^N, \quad \mathbf{m}_i[\text{domain}] = c_i. \quad (6)$$

During routing, $\mathcal{V}$ enables domain determination based on nearest-neighbor distributions. It can also be reused within domain agents to build RAG-style retrieval indexes for metallurgical question answering, so that routing and answering share consistent semantic representations, improving maintainability.

3.3.2. FAISS Index and Similar Question Retrieval

To support efficient similarity search, we build a FAISS [31] vector index. Because sentence embeddings are unit-normalized, we adopt the flat index structure `IndexFlatL2` based on ℓ_2 distance, which simplifies implementation and avoids accuracy loss introduced by quantization and compression. Given a query question q with vector $\mathbf{h} = f_\theta(q)$, we retrieve the k nearest neighbors:

$$\{i_1, \dots, i_k\} = \arg \text{topk}_i \left(-\|\mathbf{h} - \mathbf{h}_i\|_2^2 \right), \quad (7)$$

where k is the number of retrieved neighbors. In the retrieval routing stage, we typically use a small k to reduce overhead; in the LLM-based fine-grained routing stage, k can be increased to provide richer examples.

Retrieved results are encapsulated as lightweight document objects containing only text and metadata relevant to routing. This reduces inter-module data-transfer overhead and facilitates logging and analysis of routing behavior.

3.4. Multi-Stage Question Routing Mechanism

Based on the vectorized knowledge space, we design a multi-stage router composed of “ultra-fast filtering–retrieval routing–LLM-based fine-grained routing,” enabling routing decisions to achieve both high throughput and good interpretability [5–8]. From an engineering perspective, the router is implemented as an independent module that sequentially applies heuristic rules, vector-retrieval statistics, and a routing-specialized LLM [5,6]. At each stage, intermediate decisions are written into a unified routing-context structure, so that subsequent stages can reuse earlier signals and, when necessary, fall back to safe default strategies [3,6]. Algorithm 1 summarizes the complete routing procedure, including stage transitions and fallback behaviors.

Algorithm 1 consists of three stages. In Stage 1 (lines 1–7), the router preprocesses q (line 1) and performs chat-keyword matching to directly return `general_llm` when

matched (lines 2–3); otherwise it computes the normalized embedding $\mathbf{h}$ (line 4), the nearest-neighbor distance d (line 5), and returns `general_llm` if $d \geq \tau_{\text{dist}}$ (lines 6–7). In Stage 2 (lines 8–12), it retrieves the Top- k neighbors by cosine score (line 8), computes $p(c | q)$ (line 9), then obtains $\hat{c}$ and $\text{conf}(q)$ (line 10); if $\text{conf}(q) \geq \alpha$, it returns $\hat{c}$ (lines 11–12), otherwise it proceeds to Stage 3. In Stage 3 (lines 13–18), it builds a routing prompt and queries the routing LLM (line 13); if JSON parsing succeeds and the Top-1 domain belongs to $\mathcal{C}$, it returns the Top-1 predicted domain (lines 14–15), otherwise it falls back to `general_llm` (lines 16–18).

Algorithm 1 Multi-stage question routing mechanism.

Require: Query q ; steel question bank $\{(\mathbf{h}_i, c_i)\}_{i=1}^N$ with ℓ_2 -normalized vectors; domain set $\mathcal{C}$; parameters $k, \alpha, \tau_{\text{dist}}$

Ensure: Routing label $\hat{c} \in \mathcal{C} \cup \{\text{general_llm}\}$

▷ **Stage 1: Ultra-fast filtering**

```

1:  $q \leftarrow \text{Preprocess}(q)$     ▷ optional: short-query guard; strip URLs/emojis for noisy long texts
2: if  $q$  matches the chit-chat keyword set then return general_llm
3: end if
4:  $\mathbf{h} = f_{\theta}(q); \mathbf{h} \leftarrow \mathbf{h} / \|\mathbf{h}\|_2$ 
5:  $d = \min_i \|\mathbf{h} - \mathbf{h}_i\|_2$                                 ▷ Equation (8)
6: if  $d \geq \tau_{\text{dist}}$  then return general_llm
7: end if

      style="text-align: right;">▷ Stage 2: Retrieval routing
8: Retrieve Top- $k$  neighbors  $i_1, \dots, i_k$  by cosine score  $s(q, i) = \mathbf{h}^\top \mathbf{h}_i$                                 ▷ Equation (9)
9: Compute  $p(c | q)$  for  $c \in \mathcal{C}$  by Equations (10)–(11)
10:  $\hat{c} = \arg \max_{c \in \mathcal{C}} p(c | q); \text{conf}(q) = p(\hat{c} | q)$                                 ▷ Equation (12)
11: if  $\text{conf}(q) \geq \alpha$  then return  $\hat{c}$ 
12: end if

      style="text-align: right;">▷ Stage 3: LLM-based fine-grained routing
13: Build a routing prompt from  $q, \mathcal{C}$ , and Top- $k$  neighbors; query routing LLM
14: if JSON parsing succeeds and Top-1 domain  $\in \mathcal{C}$  then
15:   return Top-1 predicted domain
16: else
17:   return general_llm
18: end if
  
```

3.4.1. Security Considerations and Prompt-Injection Resilience

Although STAR focuses on routing (rather than executing actions), industrial deployment still requires basic resilience to adversarial or prompt-injection inputs. We consider two practical threat surfaces: (i) *user-side injection* where an input query attempts to override system instructions (e.g., “ignore prior rules and output domain X”) and (ii) *retrieval-side injection* where retrieved texts contain imperative instructions that could mislead an LLM-based router or downstream agents.

First, the router treats the Stage 3 LLM output as untrusted: it is accepted only when JSON parsing succeeds and the predicted domain belongs to the allowlist $\mathcal{C}$; otherwise it falls back to `general_llm`. Second, we isolate the routing prompt from user instructions by enforcing a fixed system prompt and a strictly structured output schema with temperature = 0. Third, we apply lightweight sanitization in `Preprocess(·)` to remove URLs/emojis and truncate abnormally long inputs, reducing the attack surface from long adversarial payloads. Finally, we maintain routing logs for low-confidence cases and abnormal outputs for auditing and iterative hardening.

In the router-plus-agents blueprint, retrieved documents are treated as *data*, not instructions: prompts explicitly instruct the answering model to ignore any imperative text in retrieved snippets and to ground answers in factual content only. We additionally recom-

mend (and implement in engineering) separating system prompts from retrieved content, limiting snippet length, and refusing to follow instructions that request revealing system prompts, keys, or internal policies. These measures reduce the risk that prompt injection propagates from retrieval into generation, which is especially important in high-stakes industrial environments.

3.4.2. Ultra-Fast Filtering Stage: Steel vs. General Classification

In the first stage, we determine at low computational cost whether the input is a steel-related technical query or should be directly handled by the general-purpose agent. Let the query be q . For very short queries, we adopt a conservative setting in $\text{Preprocess}(\cdot)$ to reduce false rejections caused by insufficient context. For abnormally long texts with irregular structure, we use regular-expression rules to strip URLs, emojis, and other noise that is only weakly related to professional semantics. In implementation, the above heuristics are encapsulated as an optional preprocessing function $\text{Preprocess}(\cdot)$, including a short-query guard and noise stripping (e.g., URLs/emojis) for abnormally long texts; the length thresholds are configurable.

Second, we maintain a keyword set indicative of chit-chat or everyday conversation (e.g., “weather,” “jokes,” “write a poem,” “have a chat,” etc.). If q matches this set, it is directly classified as a general query and routed to the general-purpose agent. In practice, this keyword set is maintained as a constant and iteratively updated using a small number of real dialogue samples to better match actual usage scenarios.

Finally, for queries not covered by the above rules, we compute the distance d between q and its nearest neighbor in the steel-domain question bank [5,29,30]:

$$d = \min_{i \in \{1, \dots, N\}} \|\mathbf{h} - \mathbf{h}_i\|_2, \quad (8)$$

where $\mathbf{h} = f_\theta(q)$ and both $\mathbf{h}$ and database vectors $\mathbf{h}_i$ are ℓ_2 -normalized. If $d < \tau_{\text{dist}}$, we classify q as steel-related; otherwise, it is classified as general. In our experiments, τ_{dist} is tuned on a held-out validation set; the default value in the current implementation is approximately 1.4 and can be adjusted via configuration files.

If the query is classified as general, the router immediately returns the domain label `general_llm` and terminates subsequent processing; otherwise, it proceeds to the next stage. Since a large proportion of chit-chat and non-professional queries can be filtered out at this stage, the overall number of LLM calls and the average system latency are reduced [3,5]. Concretely, the first stage follows a deterministic decision order: (i) chit-chat keyword match $\rightarrow$ `general_llm`; otherwise (ii) nearest-neighbor distance test in Equation (8). Accordingly, the router returns `general_llm` whenever $d \geq \tau_{\text{dist}}$, and proceeds to the subsequent routing stages only when $d < \tau_{\text{dist}}$.

3.4.3. Retrieval Routing Stage: Domain Determination via Nearest-Neighbor Voting

For queries classified as steel-related, the second stage performs fast domain determination based on the domain distribution of retrieved nearest neighbors.

Retrieval Score and Top- k Selection

We use ℓ_2 -normalized embeddings and define the similarity score between query q and a bank item i as cosine similarity:

$$s(q, i) = \cos(\mathbf{h}, \mathbf{h}_i) = \mathbf{h}^\top \mathbf{h}_i, \quad \|\mathbf{h}\|_2 = \|\mathbf{h}_i\|_2 = 1. \quad (9)$$

The Top- k neighbors $\{i_1, \dots, i_k\}$ are obtained by sorting $s(q, i)$ in descending order. For normalized vectors, cosine ranking is equivalent to Euclidean ranking since $\|\mathbf{h} - \mathbf{h}_i\|_2^2 = 2 - 2\mathbf{h}^\top \mathbf{h}_i$.

Let the indices of the k most similar historical questions be $i_1, \dots, i_k$, with domain labels $c_{i_1}, \dots, c_{i_k}$. We define domain counts as

$$n_c = \left| \left\{ j \mid c_{i_j} = c \right\} \right|, \quad c \in \mathcal{C}, \quad (10)$$

and compute the neighbor-induced domain distribution:

$$p(c \mid q) = \frac{n_c}{\sum_{c' \in \mathcal{C}} n_{c'}}, \quad (11)$$

note that $\sum_{c' \in \mathcal{C}} n_{c'} = k$, and therefore $p(c \mid q)$, corresponds to the empirical label proportion among the Top- k neighbors. We define the routing prediction and confidence as

$$\hat{c} = \arg \max_{c \in \mathcal{C}} p(c \mid q), \quad \text{conf}(q) = \max_{c \in \mathcal{C}} p(c \mid q) = p(\hat{c} \mid q). \quad (12)$$

If $\text{conf}(q) \geq \alpha$, we regard the query as being strongly concentrated around historical questions from that domain in semantic space and directly output $\hat{c}$ as the retrieval routing result. Otherwise, i.e., $\text{conf}(q) < \alpha$, the router abstains at this stage and triggers the subsequent LLM-based fine-grained routing stage to handle complex or boundary cases.

By default, we set $k = 5$ and $\alpha = 0.7$, i.e., the domain is selected when a clear majority of nearest neighbors come from the same domain. In our experiments, α is tuned on a held-out validation set to balance retrieval routing coverage and Stage 3 invocation.

This stage relies only on vector-retrieval and counting operations and does not invoke a large model, making it suitable for handling a large volume of typical engineering questions at low cost. Moreover, the distribution $p(c \mid q)$ supports error analysis: misrouted cases can be diagnosed by examining whether errors arise from embedding limitations, unclear domain boundaries, or an aggressive threshold choice.

3.4.4. LLM-Based Fine-Grained Routing Stage: Handling Complex/Boundary Cases

When the second-stage confidence $\text{conf}(q) = \max_{c \in \mathcal{C}} p(c \mid q)$ is below α (Equation (12)), retrieval-based voting is deemed insufficient and the router triggers the third stage: LLM-based fine-grained routing [5]. Equivalently, this stage is triggered when the second-stage routing confidence $\text{conf}(q) = \max_{c \in \mathcal{C}} p(c \mid q)$ is below α (see Equation (12)).

In this stage, a routing-specialized LLM serves as the decision-maker [3,7,8]. The retrieved similar questions with their domain labels, the candidate domain list, and the original query are organized into a routing prompt template [3,5]. Concretely, we select representative nearest-neighbor questions and format them as “question content + domain identifier.” In practice, we use the same Top- k nearest neighbors retrieved in the second stage as the representative examples. The domain set $\mathcal{C}$ is also provided as a list of domain IDs with Chinese descriptions. These inputs are fed to the routing LLM, which is asked to output a structured JSON object [3,5,7,8]:

```
{
  "top_domains": [
    {"name": "...", "score": 0.83},
    {"name": "...", "score": 0.27}
  ],
  "is_multidomain": false
}
```

here, *name* denotes a predefined domain identifier in $\mathcal{C}$. The *score* field is a relative confidence used for ranking candidate domains. We take the domain with the highest *score* in *top_domains* as the final routing label; multi-domain handling and Top-*k* combined routing are left for future extensions.

To improve reproducibility and parsing robustness, we set the decoding temperature to 0 and validate outputs using regular-expression constraints and JSON parsing [3,5]. The routing result is accepted only when JSON parsing succeeds and the Top-1 predicted domain in *top_domains* belongs to $\mathcal{C}$; otherwise it is treated as abnormal and triggers fallback. If parsing fails, if any predicted domain falls outside the predefined domain set, or if any other abnormal condition occurs, the system falls back to a conservative strategy and routes the query to the general-purpose agent *general_llm*, ensuring usable default behavior even in extreme cases [3,6].

In real plants, some questions legitimately span multiple stages (e.g., “How does ladle composition affect slab surface cracks?”). To handle such cases, STAR can be extended to produce a Top-*k* domain set instead of a single label. Concretely, when Stage 3 returns *is_multidomain*=true or when the Top-2 scores are close (e.g., $\Delta = \text{score}_1 - \text{score}_2 < 0.15$), the router outputs $\{\hat{c}_1, \hat{c}_2\}$. Downstream, two practical strategies are supported: (i) *parallel retrieval* from both domain-scoped indexes and a unified answer synthesis step that cites evidence from both domains, or (ii) *clarification-first* where the system asks a short disambiguating question when the cost of multi-agent invocation is undesirable. This extension preserves STAR’s controllability: multi-domain routing is triggered only for boundary cases, while typical single-intent queries remain on the fast path.

3.4.5. Threshold Tuning and Low-Confidence Handling

We tune τ_{dist} (Stage 1) and α (Stage 2) on a held-out validation set via grid search. For Stage 1, τ_{dist} is selected as a steel-versus-general threshold to maximize validation F1 (or under a high-recall constraint for steel queries). For Stage 2, α controls the trade-off between retrieval routing coverage and the invocation rate of Stage 3; we select α by maximizing routing accuracy while keeping the LLM call rate below a target budget. When $\text{conf}(q) < \alpha$, Stage 2 abstains and forwards the query to Stage 3; if Stage 3 produces abnormal outputs (e.g., JSON parsing failure or predicted domain not in $\mathcal{C}$), the router falls back to *general_llm*.

4. Experiments

4.1. Experimental Setup

This subsection describes the datasets, the annotation and evaluation protocols, implementation details of the router, and the evaluation metrics. We note that the experiments in this paper focus on routing performance and efficiency; evaluating end-to-end answer quality of downstream agents is beyond the current scope.

4.1.1. Datasets and Splits

We conduct routing experiments on the domain-specific question dataset $\mathcal{S} = \{(q_i, c_i)\}_{i=1}^N$ constructed in Section 3, where q_i is a natural-language question and $c_i \in \mathcal{C}$ is a fine-grained metallurgy-domain label. Since the deployment scenario involves both steel-domain technical queries and general chit-chat or everyday questions, the evaluation data consist of two parts:

We split the steel-domain questions into three disjoint subsets (train/validation/test). The steel-domain question bank used for nearest-neighbor retrieval is constructed from the training split only. Routing hyperparameters (e.g., τ_{dist} and α) are selected on the validation split, and all reported results are obtained on the held-out test split. To reduce information

leakage, we remove near-duplicate questions across splits. Non-steel questions are also split into validation/test subsets and are used together with steel questions to evaluate steel-versus-general discrimination.

4.1.2. Query Sources and Labeling Protocol

The steel-domain queries are primarily *synthetic* questions generated from OCR-extracted metallurgical text segments (Section 3) to cover long-tail engineering intents across the eight process domains. Non-steel queries are collected from general Chinese QA/dialog corpora and/or internal dialogue logs, and are labeled as `general_11m`. We explicitly distinguish *synthetic* queries from *real* user logs in all analyses to avoid overstating deployment performance.

For each generated question, we instruct the LLM to assign a domain label by comparing the predefined domain descriptions and keywords (Section 3). To reduce randomness, we use deterministic decoding (temperature = 0) and enforce format constraints on the output.

To mitigate circularity risks (LLM-generated and LLM-labeled data), we construct an *expert-validated* subset for evaluation. Specifically, we randomly sample a subset of queries from each domain and ask two domain experts to independently label them according to the same domain guideline. Disagreements are resolved by discussion or by a third senior expert as an adjudicator. We report inter-annotator agreement using Cohen’s κ (two annotators) and the final expert-labeled set is used as a reference testbed for key results.

In our expert-validated subset, the two annotators achieve an agreement of $\kappa = 0.78$. The LLM-assigned labels match the expert consensus on 89.5% of samples. Table 2 summarizes the query sources, labeling methods, and dataset sizes.

Table 2. Summary of query sources and labeling.

Subset	Source	Labeling Method	Size
Steel-domain (main)	OCR metallurgy corpus → LLM question synthesis	LLM labeling under domain definitions; JSON/schema validation + rule checks	3136
Non-steel	General QA/dialog corpora and dialog logs	Fixed label <code>general_11m</code>	≈2000
Expert-validated	Stratified sample from the steel-domain set (8 domains; 50 per domain)	3 experts: 2 independent labels; 1 adjudicates conflicts	400

- Steel-domain questions.** We use automatically generated and annotated questions from the high-quality metallurgy corpus $\mathcal{D}$, covering eight domains: raw materials and ironmaking, steelmaking and secondary refining, continuous casting, rolling, heat treatment, steel grade design, defects and quality, and production and green/low-carbon metallurgy. This subset contains approximately 3136 instances with an approximately balanced domain distribution, ensuring that each domain has sufficient samples for evaluation. We evaluate fine-grained domain routing (Top-1 accuracy and macro-F1) on the 8-way steel-domain subset only.
- Non-steel questions.** Non-steel questions (e.g., chit-chat, general writing, and everyday consulting) are sampled from real dialogue logs or open-source Chinese QA corpora. All such instances are labeled as `general_11m` and used to evaluate the steel-versus-general classification capability of the ultra-fast filtering stage. This subset contains about 2000 instances. We evaluate the steel-versus-general filtering stage as a binary classification problem on the mixed set; in deployment, predicted general queries are routed to `general_11m`.

4.1.3. Implementation Details

For embeddings, all routing-related experiments use the Chinese sentence-embedding model `bge-large-zh` as the encoding function f_θ , mapping questions into a d -dimensional vector space with ℓ_2 normalization. We build the vector index using FAISS `IndexFlatL2`. With unit-normalized embeddings; ℓ_2 distance induces an equivalent nearest-neighbor ranking to cosine similarity.

The multi-stage router follows the workflow described in Section 3: the ultra-fast filtering stage combines rules on text length, chat-keyword matching, and a nearest-neighbor distance threshold τ_{dist} to the steel-domain question bank; the retrieval routing stage determines domains based on the domain distribution of the k nearest neighbors and a threshold α .

Note that τ_{dist} and α are not learned parameters; they are hyperparameters selected on the validation set (e.g., via grid search), while the embedding model f_θ is fixed throughout all routing experiments.

For LLMs, the general-purpose agent and some domain agents use models from the Zhipu GLM series (e.g., `glm-4.5-flash`), while the routing-specialized LLM uses `glm-4-flash` with temperature set to 0 to produce deterministic structured JSON outputs. All model calls are made through a unified service interface, with timeout limits, maximum token lengths, and other parameters aligned with the deployment environment. Experiments are conducted on NVIDIA A800-SXM4-80GB GPUs.

4.1.4. Evaluation Metrics

We report the Top-1 accuracy and macro-averaged F1 for fine-grained domain routing. Let the ground-truth label of each sample in the test set be c^* and the predicted Top-1 label be $\hat{c}(q)$. Then

$$\begin{aligned} \text{Acc} &= \frac{1}{|\mathcal{S}_{\text{test}}|} \sum_{(q, c^*) \in \mathcal{S}_{\text{test}}} \mathbb{I}[\hat{c}(q) = c^*], \\ \text{Macro-F1} &= \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{F1}_c, \end{aligned} \quad (13)$$

where F1_c is the per-domain F1 score. Unless otherwise specified, Acc and macro-F1 are reported on the steel-domain test subset $\mathcal{S}_{\text{test}}$ for 8-way routing evaluation.

For steel-versus-general discrimination on the mixed test set, we report **precision (P)** (the fraction of predicted positives that are truly steel), **recall (R)** (the fraction of steel queries that are correctly predicted as positive), and **F1** (the harmonic mean of precision and recall) for both classes, as well as the overall binary accuracy (the fraction of correctly classified queries).

To characterize efficiency, we report the **Stage 3 invocation rate** (fraction of queries forwarded to Stage 3) and the **Stage 2 coverage** (fraction of queries handled by Stage 2 with $\text{conf}(q) \geq \alpha$), abbreviated as **S3 Inv.** and **S2 Cov.**, respectively.

4.2. Routing Performance Evaluation

This subsection evaluates the proposed multi-stage router in terms of fine-grained domain classification and steel-versus-general discrimination, and analyzes inter-domain confusion patterns.

4.2.1. Latency, Cost, and Failure Modes

We report average per-query latency for each stage and the overall routing pipeline under the deployment environment. Stage 1 and Stage 2 are embedding/retrieval-only operations, while Stage 3 triggers an additional LLM call. We also report the Stage 3

invocation rate, since it dominates online cost. Table 3 summarizes these deployment statistics on the mixed test set.

Table 3. Deployment statistics of the router (mixed test set).

Component	Avg. (ms)	Rate	Notes
Stage 1 (filter)	14	1.000	rules + embedding; 1-NN distance
Stage 2 (retrieval vote)	0.7	0.943	FAISS Top- k ; runs if Stage 1 predicts steel
Stage 3 (LLM refine)	650	0.283	triggered when $\text{conf} < \alpha$; main cost
Overall (end-to-end)	199	–	$t_1 + r_2 t_2 + r_3 t_3$ (avg.)

We treat the routing LLM output as untrusted and only accept it when JSON parsing succeeds and the predicted domain ID is within the allowlist $\mathcal{C}$; otherwise we fall back to `general_llm`. We log low-confidence cases and parsing failures for inspection and iterative improvement.

4.2.2. End-to-End Online Latency and Comparison to Conventional RAG

We clarify the online execution path in deployment. OCR preprocessing and corpus construction are performed *offline*; the *online* pipeline includes routing (Stages 1–3), domain-scoped retrieval (RAG), and answer generation by the selected agent. Therefore, end-to-end response time is dominated by answer generation and (when triggered) the Stage 3 refinement call.

We measure online latency on NVIDIA A800-SXM4-80GB GPU; Intel Xeon CPU; 512 GB RAM, using 1000 mixed queries. We report average latency for (i) routing-only, (ii) retrieval-only, and (iii) end-to-end routing+retrieval+generation. For a fair comparison, we implement a conventional single-RAG baseline that uses the same embedding model and retrieval backend, but *without* domain routing (i.e., one shared index and one fixed prompt).

Let t_1, t_2, t_3 denote Stage 1/2/3 latency, and let r_3 be the Stage 3 invocation rate. Let t_{rag} be retrieval time and t_{gen} be generation time. Then the expected end-to-end latency is:

$$\mathbb{E}[T] \approx (t_1 + t_2 + r_3 t_3) + t_{\text{rag}} + t_{\text{gen}}. \quad (14)$$

Since t_1 and t_2 are lightweight, STAR increases overhead mainly through the conditional Stage 3 calls, while potentially improving answer relevance via domain-scoped retrieval, as reflected in the end-to-end latency breakdown in Table 4.

Table 4. End-to-end online latency comparison.

System	Route (ms)	Retr. (ms)	Gen. (ms)	Notes
Conventional RAG (single index)	0	10	1200	shared index; shared prompt
STAR (router + domain RAG)	199	8	1210	domain-scoped index/prompt

4.2.3. Domain Routing Results

On the steel-domain test set, we evaluate the router on an 8-way fine-grained domain classification. We report Top-1 accuracy, macro-F1, and per-domain precision/recall/F1, as summarized in Table 5. We further compare our method with a retrieval-voting baseline in Section 4.3. Overall, the proposed router achieves a Top-1 accuracy of 0.921 and a macro-F1 of 0.899 on the 8-way steel-domain test set. Per-domain F1 ranges from 0.84 to 0.94 (Table 5).

Domains with more distinctive terminology and clearer process boundaries (e.g., `continuous_casting`, `rolling_control`, and `grade_design`) achieve F1 above 0.92. In contrast, `raw_ironmaking` and `heat_treatment` are relatively lower (0.84 and 0.86), mainly

due to precision/recall imbalance: *raw_ironmaking* shows high recall but lower precision, while *heat_treatment* shows high precision but lower recall when microstructure–property questions lack explicit process-stage cues.

This is consistent with real steel production, where quality and energy-efficiency issues often involve multiple stages, including composition design, metallurgical operations, and production organization.

Table 5. Per-domain precision/recall/F1 for 8-way steel-domain routing.

Domain ID	Precision	Recall	F1
<i>raw_ironmaking</i>	0.76	0.94	0.84
<i>steelmaking_refining</i>	0.85	0.93	0.89
<i>continuous_casting</i>	0.92	0.92	0.92
<i>rolling_control</i>	0.91	0.95	0.93
<i>heat_treatment</i>	0.93	0.80	0.86
<i>grade_design</i>	0.93	0.95	0.94
<i>defect_qc</i>	0.98	0.84	0.90
<i>prod_green</i>	0.97	0.86	0.91

4.2.4. Steel vs. General Routing Results

On the mixed steel/general test set (3136 steel-domain questions and 2000 non-steel/chit-chat questions), the ultra-fast filtering stage achieves a recall of about 0.999 and an F1 score of about 0.79 for the steel class. For the general class, the F1 score is around 0.25, with an overall binary accuracy of approximately 0.67, as shown in Table 6. This reflects the design trade-off: the filter prioritizes avoiding misclassifying steel-related queries as general queries, while allowing some general queries to be routed into the steel channel; such cases are then handled by subsequent retrieval and LLM-based routing stages.

Table 6. Overall metrics for 8-way routing and steel-versus-general filtering.

Level	Task/Class	Precision	Recall	F1	Accuracy
Fine-grained domain routing	8 steel domains	–	–	0.899	0.921
Fast filter (steel vs. general)	Steel (1)	0.647	0.999	0.785	–
Fast filter (steel vs. general)	General (0)	0.990	0.144	0.251	–
Fast filter (steel vs. general)	Overall (binary)	–	–	–	0.666

4.2.5. Confusion Matrix Analysis

To visualize confusion relationships between domains, we plot a normalized 8×8 confusion matrix on the steel-domain test set (Figure 2). Diagonal entries reflect per-domain accuracy, while off-diagonal entries reveal major sources of misclassification.

We observe that a portion of errors concentrates between neighboring stages along the process chain—for example, confusion between *steelmaking* and *secondary refining* and *continuous casting*, as well as misclassification between *rolling control* and *heat treatment* when queries involve microstructure–property issues. These patterns suggest that routing accuracy may be further improved by explicitly modeling upstream–downstream process relationships or introducing constraints derived from process flow diagrams.

4.2.6. Robustness to Short, Ambiguous, and Multi-Intent Queries

Industrial queries are often short, underspecified, or contain multiple intents. To assess robustness, we construct three diagnostic subsets from the test set: (i) *short queries* with length ≤ 12 characters, (ii) *ambiguous queries* where Stage 2 confidence $\text{conf}(q) < \alpha$, and (iii) *multi-intent queries* identified by simple conjunction patterns or manual tags (e.g., 同

时/以及/并且/对比/一方面...另一方面...). We report routing accuracy/macro-F1 and the Stage 3 invocation rate on each subset in Table 7. This analysis reveals how STAR performs under short, ambiguous, and multi-intent queries, when it relies more on LLM refinement, and where additional disambiguation or multi-domain routing may be needed.

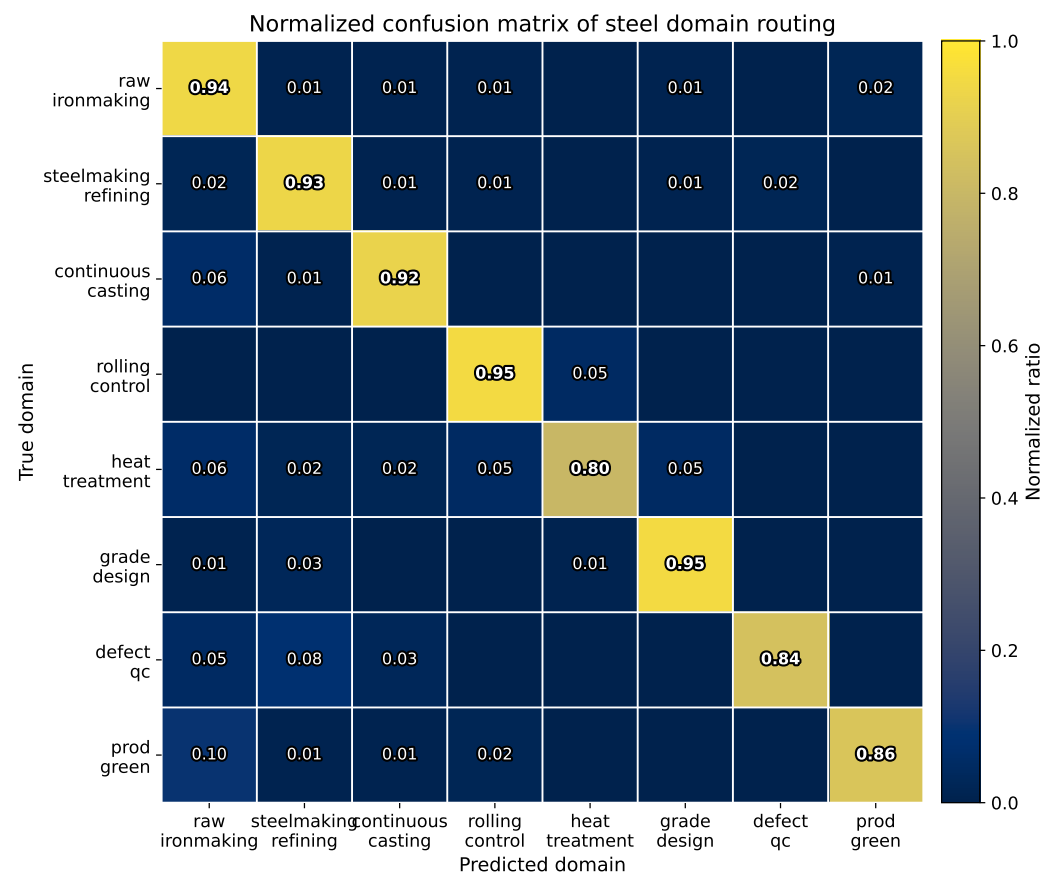


Figure 2. Normalized confusion matrix of the proposed multi-stage router on the steel-domain test set. Diagonal entries denote per-domain accuracy, and off-diagonal entries show the main confusion patterns between neighboring process stages.

Table 7. Robustness analysis on diagnostic subsets.

Subset	Size	Top-1 Acc.	Macro-F1	S3 Inv.	Notes
Short queries (≤ 12 chars)	620	0.890	0.870	0.420	underspecified intents
Ambiguous (Stage 2 conf $< \alpha$)	940	0.880	0.860	1.000	Stage 3 frequently triggered
Multi-intent (heuristic/manual)	280	0.840	0.810	0.680	may require multi-domain routing

4.3. Baseline Comparison

We compare the proposed multi-stage router with representative baselines, including embedding-only routing, a simple supervised classifier on embeddings, and the LLM-only classification, following the evaluation protocol in Section 4.1.

We include a nearest-neighbor (NN, $k = 1$) baseline that assigns each query to the domain label of its closest question-bank entry in embedding space. This embedding-only NN baseline achieves a Top-1 accuracy of 0.848 and a macro-F1 of 0.690 on the 8-way steel-domain test set, indicating limited robustness when relying on a single nearest neighbor without an explicit decision boundary.

We use the retrieval-only voting baseline (*Retrieval-only*) that retrieves the Top- k nearest neighbors and predicts the most frequent domain among them. With $k = 5$, retrieval-only voting improves Top-1 accuracy to 0.871 with a macro-F1 of 0.725, suggesting that

aggregating multiple neighbors can stabilize predictions compared to $k = 1$, while the per-class performance remains less uniform.

We train a lightweight logistic regression (LR) classifier on fixed sentence embeddings $\mathbf{h} = f_{\theta}(q)$. This supervised LR baseline attains 0.907 Top-1 accuracy and 0.853 macro-F1, showing that a learned linear classifier on embeddings substantially improves class-balanced performance over purely similarity-based routing.

We query the routing of LLM to directly predict the domain from q and $\mathcal{C}$, without providing retrieved neighbors. The LLM-only baseline reaches 0.832 Top-1 accuracy and 0.713 macro-F1, which is lower than embedding-based routing, suggesting that the model benefits from domain-specific evidence rather than relying solely on label descriptions.

Table 8 summarizes the comparison on the 8-way steel-domain test set. Overall, the proposed multi-stage router achieves a Top-1 accuracy of 0.921 and a macro-F1 of 0.899, outperforming all baselines. In particular, compared with retrieval-only voting, our router improves Top-1 accuracy from 0.871 to 0.921 and macro-F1 from 0.725 to 0.899. The stronger macro-F1 indicates more consistent performance across the eight domains, consistent with the design of combining fast filtering, retrieval evidence, and multi-stage decision rules.

Table 8. Baseline comparison on 8-way steel-domain routing.

Method	Top-1 Acc.	Macro-F1
Embedding-only: NN ($k = 1$)	0.848	0.690
Retrieval-only: voting ($k = 5$)	0.871	0.725
Supervised: LR on embeddings	0.907	0.853
LLM-only (no retrieval examples)	0.832	0.713
Proposed multi-stage router	0.921	0.899

4.4. Ablation Study

To quantify the contribution of each stage in Algorithm 1, we conduct an ablation study by removing or simplifying individual components while keeping the remaining settings unchanged. Unless otherwise specified, all ablations use the same embedding model, the same retrieval bank, and the same evaluation protocol as in Section 4.1.

4.4.1. Stage 1 Ablations: Steel-Versus-General Filtering

Stage 1 combines a chit-chat keyword filter and a nearest-neighbor distance threshold for steel-versus-general discrimination. We evaluate two variants: (i) removing the chit-chat keyword filter and (ii) removing the distance threshold test. We report **Steel FNR** (false negative rate) and **General FPR** (false positive rate) on the mixed test set, where Steel FNR measures the fraction of steel queries misrouted to `general_llm`, and General FPR measures the fraction of general queries routed into the steel channel. Table 9 reports the error rates for these variants.

Table 9. Ablation ranges of Stage 1 on steel-versus-general discrimination.

Variant	Steel FNR	General FPR
Stage 1 (full)	0.001	0.856
w/o chit-chat keyword filter	0.003	0.930
w/o distance threshold (τ_{dist})	0.002	0.995

4.4.2. Stage 2/3 Ablations: Confidence Thresholding and LLM Refinement

Stage 2 produces a domain prediction and a confidence score from the neighbor-labeled distribution and forwards low-confidence cases to Stage 3 for LLM-based refinement. To assess the role of Stage 3, we evaluate *w/o Stage 3 (Stage 2 only)*, which disables the LLM

refinement and always returns the Stage 2 prediction. To assess the role of retrieved examples in Stage 3, we further evaluate *Stage 3 w/o neighbors*, where the routing LLM is queried using only q and the domain list $\mathcal{C}$. We report Top-1 accuracy and macro-F1 on the 8-way steel-domain test set (Table 10). In addition, we report **S2 Cov.** and **S3 Inv.** to reflect the accuracy–cost trade-off.

Table 10. Ablation results on 8-way steel-domain routing.

Variant	Top-1 Acc.	Macro-F1	S2 Cov.	S3 Inv.
Proposed router (full)	0.921	0.899	0.700	0.300
Stage 2 only (no Stage 3)	0.871	0.725	1.000	0.000
Stage 3 w/o retrieved neighbors	0.895	0.830	0.700	0.300

4.4.3. Hyperparameter Sensitivity

We examine the sensitivity of the router to key hyperparameters in Algorithm 1 by sweeping one parameter at a time and evaluating the 8-way steel-domain routing task.

Stage 1: distance threshold τ_{dist} .

We sweep $\tau_{\text{dist}} \in \{1.2, 1.3, 1.4, 1.5\}$. As shown in Figure 3a, macro-F1 improves as τ_{dist} increases from 1.2 to 1.4, and then remains stable. This indicates that a moderately strict distance threshold effectively filters out non-domain queries while avoiding overly aggressive rejections.

Stage 2: confidence threshold α .

We sweep $\alpha \in \{0.3, 0.5, 0.7, 0.9\}$. Figure 3b shows that $\alpha = 0.7$ gives the best performance among the tested values. Larger α makes the router more conservative and increases Stage 3 invocations, but does not consistently improve routing quality.

Stage 3: Top- k neighbors.

We sweep $k \in \{1, 3, 5, 7\}$. As shown in Figure 3c, macro-F1 increases from $k = 1$ to $k = 5$ and slightly drops at $k = 7$, suggesting that a moderate neighborhood size balances stability and noise in label aggregation.

Across these sweeps, performance is relatively stable within the tested ranges. We use $\tau_{\text{dist}} = 1.4$, $\alpha = 0.7$, and $k = 5$ in all experiments.

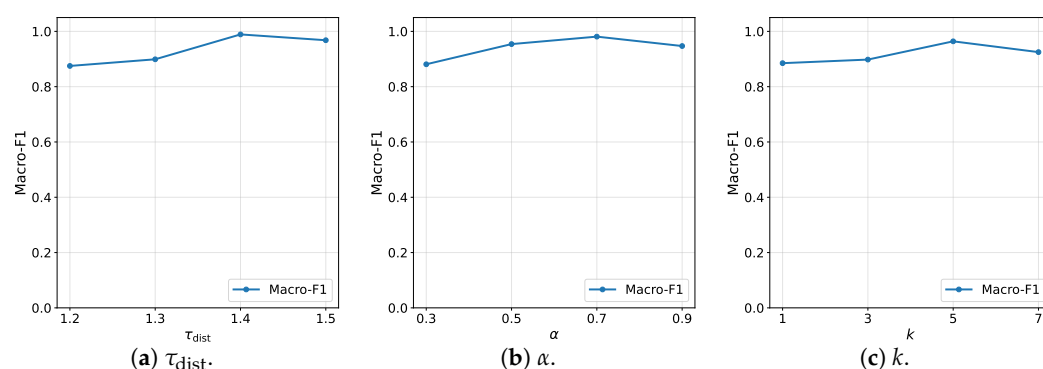


Figure 3. Sensitivity analysis of key hyperparameters with four settings for each hyperparameter.

5. Conclusions

Focusing on iron and steel metallurgy as a representative process-industry scenario, this paper proposes and implements an industrial process-domain query routing framework based on a “multi-stage routing” design, and provides a router-plus-agents integration

blueprint that maps routing labels to domain-scoped retrieval and prompting configurations. Starting from scanned textbooks, monographs, and papers, we construct a metallurgy corpus with OCR-based quality control and automatically generate a domain-labeled question dataset covering eight fine-grained domains: raw materials and ironmaking, steelmaking and secondary refining, continuous casting, rolling, heat treatment, steel grade design, defects and quality, and production and green/low-carbon metallurgy. We further build a domain-labeled vectorized knowledge space using a Chinese sentence-embedding model and FAISS, and design a multi-stage router—comprising ultra-fast filtering, retrieval-based routing, and LLM-based refinement—that combines heuristic rules, nearest-neighbor statistics, and LLM reasoning. By decoupling routing from answer generation, the proposed framework enables stage-aware query dispatching and provides a practical foundation for integrating specialized downstream components in industrial LLM systems.

Experimental results show that the proposed router significantly outperforms baselines on fine-grained domain routing, and achieves high recall for steel-domain queries as well as practically acceptable performance on the steel-versus-general discrimination task. While maintaining routing quality for complex long-tail queries, the framework effectively controls online inference cost.

This work also has several limitations and opportunities for improvement. First, the current domain partitioning and routing decisions primarily rely on a vector space constructed from static corpora and heuristic thresholds, which may still cause misrouting for cross-process-chain queries and data-sparse domains. Second, the system has not yet fully exploited structured information such as process flow diagrams, production data, and mechanistic models, and the modeling of inter-domain relationships remains relatively coarse. In addition, the impact of OCR noise and highly colloquial on-site expressions on embedding quality and routing stability warrants more systematic investigation.

Future work will proceed in four directions: (1) combining active learning with online feedback to continuously expand the domain question bank for long-tail queries, and exploring lightweight supervised routers layered on top of the current retrieval routing framework to improve adaptability to complex scenarios; (2) introducing process flow diagrams and knowledge graphs to explicitly model upstream–downstream relationships between domains, and using them to jointly constrain the decision space in both routing and answering; (3) integrating mechanistic models and real-time production data into selected domain agents, moving toward a “data–knowledge–mechanism” collaborative decision-support system; and (4) transferring the proposed approach to other process industries (e.g., non-ferrous metallurgy, chemical engineering, and cement) to validate the generality and applicability of the multi-stage routing and multi-agent collaboration framework in broader industrial settings.

Author Contributions: Conceptualization, Q.Z. and L.W.; methodology, Q.Z., W.L. and C.H.; software, C.H., S.W., F.M. and M.L.; validation, H.Z., F.M. and M.L.; formal analysis, H.Z., F.M. and M.L.; investigation, Q.Z., W.L., C.H. and S.W.; resources, Q.Z.; data curation, C.H., S.W. and M.L.; writing—original draft preparation, W.L. and Q.Z.; writing—review and editing, Q.Z., C.H., S.W. and L.W.; visualization, M.L. and S.W.; supervision, L.W. and H.Z.; project administration, L.W. and H.Z.; funding acquisition, Q.Z.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Science and Technology Major Project “Intelligent Manufacturing Systems And Robots” (No.2025ZD1602500).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request due to privacy concerns.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ghosh, A.; Chatterjee, A. *Ironmaking and Steelmaking: Theory and Practice*; PHI Learning: New Delhi, India, 2008.
2. Merten, D. Decision Support Systems for Steel Production Planning—State of the Art and Open Questions. In *Steel 4.0: Digitalization in Steel Industry*; Uygun, Y., Özgür, A., Hütt, M.T., Eds.; Springer: Cham, Switzerland, 2024; pp. 73–83.
3. Zhao, W.X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. A Survey of Large Language Models. *arXiv* **2023**, arXiv:2303.18223.
4. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models Are Few-Shot Learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
5. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; tau Yih, W.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv* **2021**, arXiv: 2005.11401.
6. Wu, T.; Li, J.; Bao, J.; Liu, Q. Language model-driven multi-agent systems for improving production efficiency and reducing carbon emissions in manufacturing. *Comput. Ind. Eng.* **2025**, *207*, 111299. <https://doi.org/10.1016/j.cie.2025.111299>.
7. Jacobs, R.A.; Jordan, M.I.; Nowlan, S.J.; Hinton, G.E. Adaptive Mixtures of Local Experts. *Neural Comput.* **1991**, *3*, 79–87.
8. Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In Proceedings of the International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
9. Ding, Y.; Luo, S.; Dai, Y.; Jiang, Y.; Li, Z.; Martin, G.; Peng, Y. A Survey on MLLM-based Visually Rich Document Understanding: Methods, Challenges, and Emerging Trends. *arXiv* **2025**, arXiv:2507.09861.
10. Zhang, Z.; Zhang, Y.; Liang, Y.; Xiang, L.; Zhao, Y.; Zhou, Y.; Zong, C. From Chaotic OCR Words to Coherent Document: A Fine-to-Coarse Zoom-Out Network for Complex-Layout Document Image Translation. In Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025), Abu Dhabi, United Arab Emirates, 19–24 January 2025; pp. 10877–10890.
11. Li, Y.; Zhao, H.; Jiang, H.; Pan, Y.; Liu, Z.; Wu, Z.; Shu, P.; Tian, J.; Yang, T.; Xu, S.; et al. Large language models for manufacturing. *arXiv* **2024**, arXiv:2410.21418.
12. Zhang, C.; Zhou, G.; Liu, Y.; Zhou, G.; Zeng, K.; Chang, F.; Ding, K. A survey on potentials, pathways and challenges of large language models in new-generation intelligent manufacturing. *Robot. Comput.-Integr. Manuf.* **2025**, *92*, 102883.
13. Jiang, T.; Zhu, D.; Wu, H.; Mao, X. Large language models empowering the steel industry: Technology and application outlook. *Yejin Zidonghua* **2025**, *49*, 1–17. (In Chinese)
14. Du, K.; Yang, B.; Xie, K.; Dong, N.; Zhang, Z.; Wang, S.; Mo, F. LLM-MANUF: An integrated framework of fine-tuning large language models for intelligent decision-making in manufacturing. *Adv. Eng. Inform.* **2025**, *65*, 103263.
15. Chandrasekhar, A.; Chan, J.; Ogoke, F.; Ajenifujah, O.; Barati Farimani, A. AMGPT: A large language model for contextual querying in additive manufacturing. *Addit. Manuf. Lett.* **2024**, *11*, 100232. <https://doi.org/10.1016/j.addlet.2024.100232>.
16. Khan, M.T.; Chen, L.; Feng, W.; Moon, S.K. Large language model-powered decision support for a metal additive manufacturing knowledge graph. *arXiv* **2025**, arXiv:2505.20308.
17. Fan, H.; Fan, Z.; Liu, C.; Zhu, J.; Gibbs, T.; Fuh, J.Y.H.; Lu, W.F.; Li, B. MetalMind: A knowledge graph-driven human-centric knowledge system for metal additive manufacturing. *npj Adv. Manuf.* **2025**, *2*, 25.
18. Li, S.; Corney, J. MechRAG: A multimodal large language model for mechanical engineering. *Commun. Eng.* **2025**, *4*, 187.
19. Fu, T.; Liu, S.; Li, P. Intelligent smelting process management system: Efficient and intelligent management strategy by incorporating large language model. *Front. Eng. Manag.* **2024**, *11*, 396–412.
20. Zhang, H.; Gu, J.; Sun, Y.; Zheng, Q.; Li, M. StiBench: An understanding benchmark for large language models in the steel metallurgy domain. *Yejin Zidonghua* **2025**, *49*, 102–111. (In Chinese) <https://doi.org/10.3969/j.issn.1000-7059.2025.04.20250162>.
21. Hu, Q.J.; Bieker, J.; Li, X.; Jiang, N.; Keigwin, B.; Ranganath, G.; Keutzer, K.; Upadhyay, S.K. RouterBench: A benchmark for multi-LLM routing system. *arXiv* **2024**, arXiv:2403.12031.
22. Ong, I.; Almahairi, A.; Wu, V.; Chiang, W.L.; Wu, T.; Gonzalez, J.E.; Kadous, M.W.; Stoica, I. RouteLLM: Learning to route LLMs with preference data. *arXiv* **2024**, arXiv:2406.18665.
23. Jitkrittum, W.; Narasimhan, H.; Rawat, A.S.; Juneja, J.; Wang, C.; Wang, Z.; Go, A.; Lee, C.Y.; Shenoy, P.; Panigrahy, R.; et al. Universal model routing for efficient LLM inference. *arXiv* **2025**, arXiv:2502.08773.
24. Fu, T.; Ge, Y.; You, Y.; Liu, E.; Yuan, Z.; Dai, G.; Yan, S.; Yang, H.; Wang, Y. R2R: Efficiently navigating divergent reasoning paths with small-large model token routing. *arXiv* **2025**, arXiv:2505.21600.
25. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv* **2023**, arXiv:2308.08155.

26. Wu, T.; Li, J.; Bao, J.; Liu, Q. ProcessCarbonAgent: A large language models-empowered autonomous agent for decision-making in manufacturing carbon emission management. *J. Manuf. Syst.* **2024**, *76*, 429–442.
27. Li, M.; Wang, R.; Zhou, X.; Zhu, Z.; Wen, Y.; Tan, R. ChatTwin: Toward Automated Digital Twin Generation for Data Center via Large Language Models. In Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '23), Istanbul Turkey, 15–16 November 2023.
28. Yang, J.; Li, S.; Wang, X.; Lu, J.; Wu, H.; Wang, X. DeFACT in ManuVerse for parallel manufacturing: Foundation models and parallel workers in smart factories. *IEEE Trans. Syst. Man, Cybern. Syst.* **2023**, *53*, 2188–2199.
29. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, 3–7 November 2019; pp. 3982–3992.
30. Douze, M.; Guzhva, A.; Deng, C.; Johnson, J.; Szilvasy, G.; Mazaré, P.E.; Lomeli, M.; Hosseini, L.; Jégou, H. The Faiss Library. *arXiv* **2024**, arXiv:2401.08281.
31. Johnson, J.; Douze, M.; Jégou, H. Billion-scale similarity search with GPUs. *arXiv* **2017**, arXiv:1702.08734.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.