Automated Construction of Job–Skill Knowledge Graphs Based on Multi-Agent Consensus and Four-Layer Verification

Lusheng Sun, Hangyu Zhou, Mengfei Chen, Mengfan Song, Qi Wang*, Hengxu Guo, Jingyuan Yang, Qihang Zhao, Ruian Yan

School of Artificial Intelligence and Big Data, Henan University of Technology, Zhengzhou, China

**Corresponding Author*

Abstract: To construct reliable job–skill knowledge graphs from heterogeneous recruitment texts, this study combines multi-agent consensus with a four-layer verification cascade. Three large language models from different vendors (iFlytek Spark, DeepSeek, Zhipu GLM-4-Flash) first annotate each JD independently under distinct cognitive roles, then converge through peer review and voting; uncertain skills are subsequently checked against knowledge graph lookup, ESCO alignment, web search, and manual review. Evaluated on 3,942 real-world JDs, the method attains an F1-score of 89.0%, improving by 4.2 points over the best single model and 13.5 points over traditional NER; the cascade reduces the hallucination rate from 15.0% to 1.4%. The ESCO alignment rate of 71.5% leaves room for improvement in covering indigenous Chinese technologies, yet the framework offers a practical pathway for occupational skill standardization and intelligent talent services.

Keywords: Knowledge Graph Construction; Multi-agent Consensus; Large Language Model; Entity Extraction; ESCO International Standard; Job Skills

1. Introduction

Recruitment platforms now hold a vast stock of job descriptions (JDs), and these texts carry authentic, up-to-date occupational skill information; therefore, it is hardly surprising that they have become a natural data source for building job–skill knowledge graphs [1].

Economic digitalization has accelerated the evolution of occupational skill requirements, as emerging technologies such as artificial intelligence, big data, and cloud computing continuously reshape employer expectations of talent. In this context, JDs on recruitment platforms serve as one of the most timely and reliable signals of the labor market's actual skill demands. Yet, the volume, heterogeneous formats, and rapid update cycles of these texts render traditional manual analysis impractical, necessitating automated methods for large-scale skill extraction and dynamic tracking.

Building high-quality knowledge graphs from multi-source heterogeneous JD texts remains non-trivial, with two principal challenges. The first concerns extraction reliability: conventional approaches rely on named entity recognition (NER) or a single language model [2], yet JDs contain abundant implicit skills, synonyms, and ambiguous expressions; therefore, single-model extraction exhibits systematic bias and random fluctuation—and models from different vendors may diverge markedly on identical inputs. The second concerns verification: existing studies typically resort to manual sampling [3], which becomes prohibitively expensive at the scale of hundreds of JDs and thousands of skills, and manual judgment is itself subject to subjective bias.

To address these issues, this paper develops an automated method that constructs job–skill knowledge graphs from multi-agent consensus combined with four-layer verification. The key distinction from prior work lies in organizing multiple LLMs from different vendors into a peer collaborative system: models first reach consensus through independent extraction, peer review, and voting, after which an automated four-layer cascade—knowledge graph lookup, ESCO alignment, web search, and manual review—verifies the results. The main contributions are as follows.

First, a cross-vendor multi-agent consensus extraction framework: three domestic large language models are assigned differentiated cognitive roles and converge through a three-stage mechanism of independent extraction, peer collaboration, and voting classification (embedded in a four-stage workflow with

external verification), which reduces the random bias and systematic blind spots of any single model while supporting an autonomous and controllable AI infrastructure. Second, a four-layer progressive verification mechanism that hierarchically confirms skill authenticity through knowledge graph lookup, ESCO international alignment, web search corroboration, and manual review, with the ESCO layer leveraging LLM translation for cross-lingual matching. Third, an end-to-end implementation covering data collection, skill extraction and verification, knowledge graph storage, and visualization, supporting downstream applications such as evolution tracking and job–person matching.

2. Related Work

2.1 Knowledge Graph Construction Method

Knowledge graph construction is usually classified into three families: rule-based, statistical learning-based, and pre-trained language model-based. Surveys by He et al. [1] and Zhang et al. [4] provide a systematic account of LLM-enhanced construction. Among the classical techniques, BiLSTM-CRF remains a mainstream choice for named entity recognition [5, 6], attaining F1-scores of 69%–87% in narrow domains, although only with substantial amounts of labeled data. Yang et al. [2] introduced a BERT-BiGRU-Attention-CRF model for job entity recognition in the digital industry, and Zhu et al. [7] probed the capability boundaries of LLMs in knowledge graph construction, proposing the AutoKG framework. What these efforts share, however, is that none of them systematically grapples with automated verification of extraction results.

2.2 Skill Extraction Techniques

In the recruitment domain, skill extraction research has been divided into two technical routes. The former relies on dictionary-based matching, whose recall is capped by dictionary coverage; the latter uses sequence labeling or generation-based machine learning. Yu et al. [8] built a job knowledge graph from training evaluation standards, and Wang et al. [9] proposed the KG-AMM model, which couples knowledge graphs with graph attention networks for job–person matching. A common weakness runs through both lines of work: skill extraction is treated as a deterministic, single-model task, so the uncertainty inherent in model outputs is never systematically considered.

2.3 Multi-Agent Methods

LLM-driven multi-agent systems have recently become the focus of intense interest. ReConcile [10], for example, seated different LLMs around a round-table conference and used weighted voting to improve reasoning tasks by 11.4%. Chain of Agents [11] adopted a worker-manager two-stage framework, gaining up to 10% on long-context tasks. In information extraction specifically, CROSSAGENTIE [12] introduced a cross-type debate mechanism, ARIS [13] blended reflective consensus with sequence taggers, and CooperKGC [14] devised a decentralized expert collaboration network. Iterative consensus has also paid off elsewhere: Kobayashi et al. [15] pushed relation precision from 79% to 97%. On a broader scale, Pan et al. [16] reviewed the two-way strengthening of LLMs and knowledge graphs, while CONSENSAGENT [17] drew attention to the sycophancy effect in multi-agent debate and responded with an anti-sycophancy protocol.

Table 1 compares the proposed method with representative works along four dimensions: multi-model type, automated verification, data scale, and system completeness.

Table 1. Comparison of the Proposed Method with Existing Works

Method	Model type	Verification	Data scale	Performance	End-to-end
BiLSTM-CRF [2]	Single NER	None	15,980	F1=85.5%	No
ReConcile [10]	Cross-vendor 3 models	None	—	Reasoning ↑11.4%	No
Chain of Agents [11]	Same-model multi Worker	None	—	Long-context ↑10%	No
CROSSAGENTIE [12]	Same-model multi Agent	None	—	Zero-shot SOTA	No
CooperKGC [14]	Same-model expert Agent	None	—	KG construction SOTA	No
Ours	**Cross-vendor 3 models**	**Four-layer**	**3,942**	**F1=89.0%**	**Yes**

Taken together, the existing works share two weaknesses: (1) most simulate "multi-agent" behavior with different parameters or roles of the same model and thereby fail to exploit cross-vendor complementarity—in Table 1, only ReConcile adopts a cross-vendor strategy; (2) systematic automated verification is missing from all of them, with most relying instead on

manual sampling.

In summary, three points distinguish our approach from prior art: (1) we use three genuinely independent LLM instances from different vendors as peer agents, not parameter variants of one model; (2) an automated four-layer verification mechanism—with the international standard ESCO included—is introduced; (3) the entire pipeline, from data collection to application services, is implemented end to end.

3. System Architecture

3.1 Overall Design

The overall design follows a three-layer separated pattern. From bottom to top, these are the infrastructure layer, the business engine layer, and the interface service layer (Figure 1). A strict one-way dependency holds across the three layers: upper layers may call lower ones but never the reverse; therefore, each layer can be swapped out and tested independently.

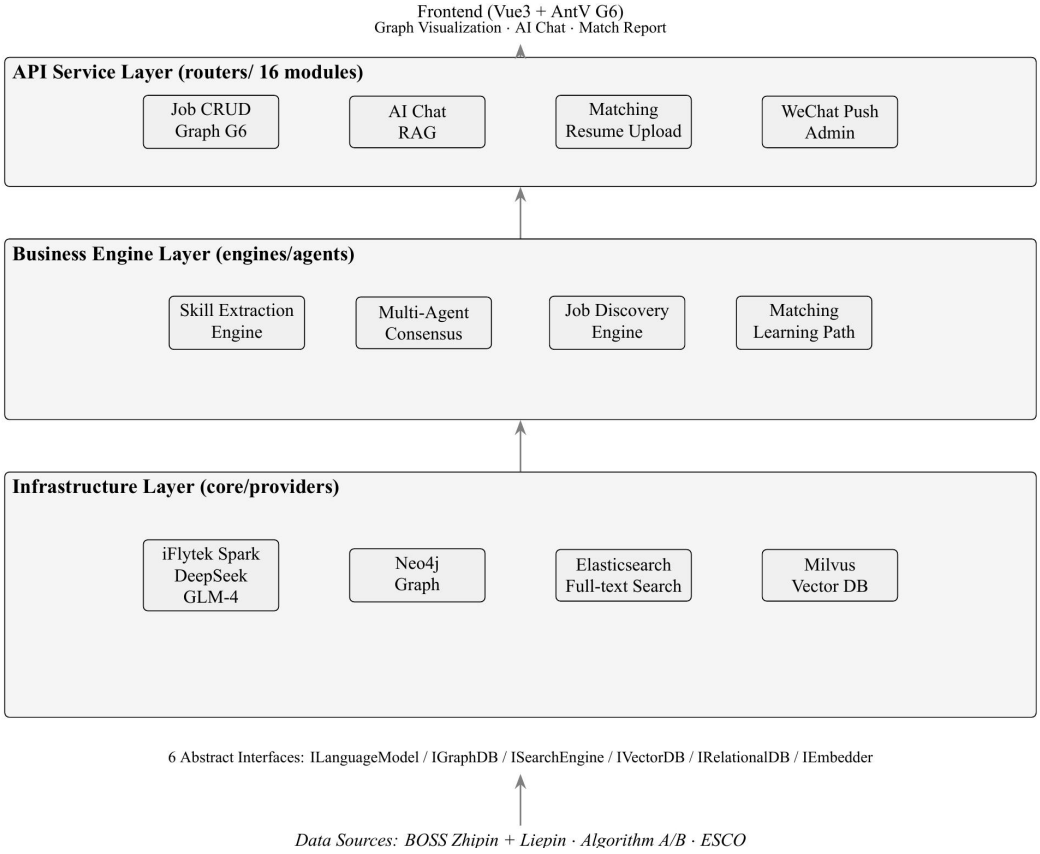

Figure 1. Overall Three-Layer System Architecture

At the bottom sits the infrastructure layer, which declares six abstract interfaces: ILanguageModel, IGraphDB, ISearchEngine, IVectorDB, IRelationalDB, and IEmbedder. Upper-level modules depend exclusively on these interfaces, never on concrete implementations. At startup, a dependency injection container instantiates the concrete classes according to environment variables; thus, switching an underlying vendor—say, swapping the LLM from iFlytek Spark to DeepSeek—amounts to editing the configuration file.

The business engine layer bundles the core capabilities of the system. A skill extraction engine invokes the LLM for structured entity extraction; a multi-agent consensus engine choreographs the three models' parallel extraction, collaborative review, and voting; a job discovery engine sifts through massive JD volumes to flag emerging jobs; a matching engine computes job–person compatibility and produces learning path recommendations; and a four-layer verification toolset checks skill authenticity at successive levels of rigor.

The interface service layer, built on FastAPI, exposes 16 RESTful API modules: job CRUD, graph querying and G6 force-directed visualization, an AI career advisor dialogue (RAG-enhanced), resume upload and parsing, and WeChat push services. All these interfaces are browsable and testable from the frontend through Swagger UI.

3.2 Data Model

The relational store holds 10 tables, three of which are central: the job definition table (hr_positions), the skill taxonomy table (hr_skill_taxonomy), and the job–skill association table (hr_position_skills). The skill taxonomy adopts a three-level tree structure (root category → subcategory → specific skill); each record carries an ESCO code reference, a normalized name, an alias list, a trend direction (RISING/DECLINING/STABLE/NEW), and annual frequency statistics. A dedicated skill history table (hr_skill_history) logs every skill change event as a quintuple (job, skill, change type, observation date, evidence), yielding traceable evolution time-series. The remaining tables cover user profiles, matching records, collection task logs, practice records, learning paths, and WeChat push logs.

Storage is deliberately distributed: Neo4j maintains the job–skill bipartite graph, Elasticsearch builds an inverted index over the raw JD texts, and Milvus holds the embedding vectors used for semantic retrieval.

4. Multi-Agent Consensus Skill Extraction and Four-Layer Verification Method

4.1 Problem Formulation

Formally, given a job description text D, skill extraction seeks a set of skill entities $S = \{(n_i, t_i, c_i)\}_{i=1}^{k}$ drawn from D, where n_i is the skill name.

$t_i \in \{SKILL,KNOWLEDGE,TOOL,SOFT\}$ is the entity type and $c_i \in [0,1]$ the confidence. Accuracy, however, is not really the crux of the matter; credibility is. Different vendor models produce output differences that are far from negligible on the same input, and resampling a single model repeatedly cannot remove its systematic bias, so a multi-model collaboration mechanism is needed to quantify and reduce this uncertainty.

4.2 Cross-Vendor Three-Model Design

We selected the three agent models with a deliberate focus on domestic Chinese large language models, motivated by the need for autonomous and controllable AI infrastructure in real-world deployments. In a pilot study, we benchmarked several mainstream domestic LLMs—iFlytek Spark, DeepSeek, GLM-4, Baichuan, and Kimi—on 50 labeled JDs, weighing both annotation accuracy and API cost. Balancing performance against affordability, we chose iFlytek Spark 4.0 Ultra as the primary model, with DeepSeek-Chat and Zhipu GLM-4-Flash as the two auxiliary agents; together, they achieved the highest F1 among all candidates and offered complementary coverage of the JD text. We deliberately chose three agents from different vendors rather than using three instances of the same model, since cross-vendor diversity better exposes the genuinely uncertain regions of extraction—each vendor's model has its own blind spots, and their disagreements mark exactly where a human reviewer should look.

Viewed through the lens of ensemble learning, multi-agent consensus only works if the individual agents are genuinely diverse. Ensemble diversity theory tells us that combining models with complementary errors trims overall variance and systematic bias beyond what any one model manages on its own. For large language models, diversity comes from three sources: the training corpora, the alignment strategies, and the decoding behavior. Cross-vendor selection maximizes the first two, whereas temperature variation within a single model merely perturbs decoding, so the outputs remain highly correlated and carry little complementary information.

Each model is given a distinct cognitive role, and the system prompts inject differentiated annotation tendencies accordingly.

Strict reviewer (strict): "Only annotate skills explicitly mentioned in the JD. Do not infer, associate, or supplement." Temperature $T=0.0$, which yields the most certain results.

Balanced judge (balanced): "Annotate explicitly mentioned skills as well as implicit but reasonable skills." Temperature $T=0.2$, so it sits between strictness and leniency.

Curious explorer (curious): "In addition to explicitly mentioned skills, pay attention to implied technology stack associations in the JD. Mark inferred skills with [?]." Temperature $T=0.4$, allowing it to capture marginal information.

The idea, importantly, is not to have the models "perform the same task three times." Rather, we deliberately engineered diversity from a cognitive perspective: the strict role holds false positives in check, the curious role guards against false negatives, and the balanced role supplies an anchor reference.

The temperature settings came out of pilot tuning. On 50 labeled JDs we ran a grid search over temperature triples, optimizing F1-score and the diversity index jointly, and settled on T=(0.0,0.2,0.4) , which delivered the best F1-score (87.3%) along with the highest diversity (0.31). Pushing temperatures higher does boost diversity, but it also inflates the explorer's false-positive rate from 12% to 23% and drags overall F1 down to 83.5%—evidence that the added noise outweighs the diversity gain.

4.3 Four-Phase Consensus Process

We settled on the four-phase design after an earlier, simpler two-stage prototype produced too many unresolved disagreements to be reliable in practice. The final pipeline separates parallel extraction from collaborative review, voting, and external verification, so that each stage can be independently validated.

The consensus mechanism runs through four phases, executed sequentially (Figure 2).

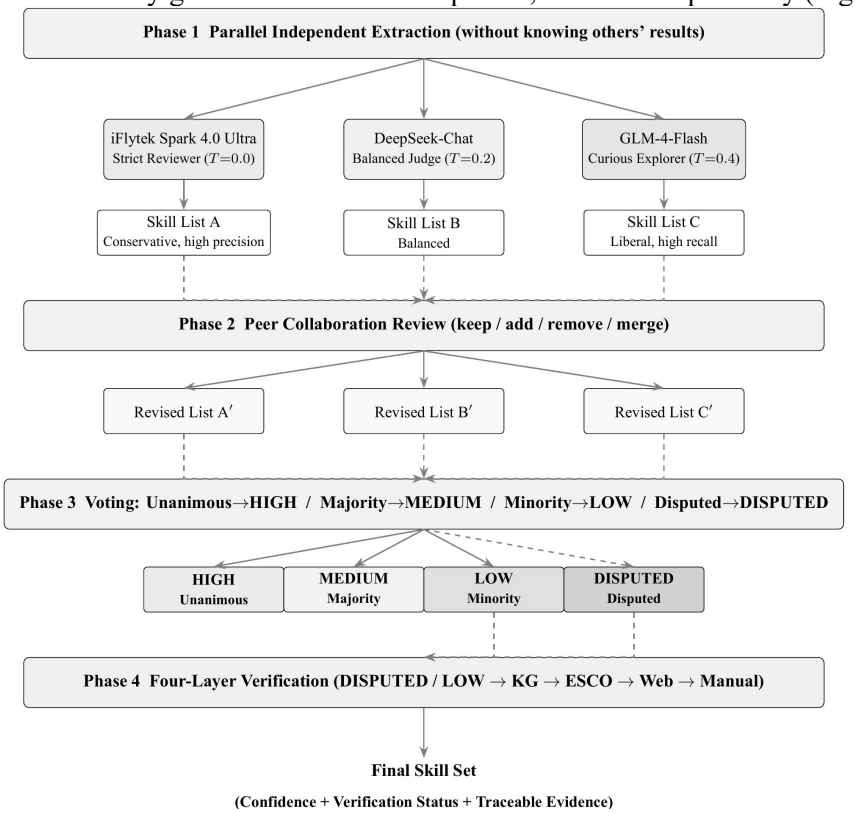

Figure 2. Four-Phase Multi-Agent Consensus Process

Phase 1: Parallel Independent Extraction. The three agents receive the same JD text at the same time and carry out extraction on their own, unaware of one another's results; the only differences lie in the role descriptions inside the system prompts. Each agent outputs an independent skill list$A$,B,C.

Phase 2: Peer Collaboration Review. The three independent results are pooled and sent back to every agent, who then autonomously applies four operations while consulting peers' judgments: keeping, adding (requiring evidence in the original text), removing, and merging. The collaboration prompt stresses that "do not blindly follow majority opinions; every skill must have evidence in the original text." Working in parallel, the three agents revise their outputs into skill listsA',B',C'.

Phase 3: Voting Classification. After the corrected results are aggregated, a consensus ratio $r(s) = |\{p \in P : s \in S_p\}| / |P|$ is computed for every skill that at least one agent annotated; this ratio fixes the confidence level: r=1.0 is HIGH, $0.5 \le r < 1.0$ is MEDIUM, and $0 < r < 0.5$ is LOW. Skills dropped in Phase 2 are labeled DISPUTED and proceed to the next verification phase.

Phase 4: External Verification. DISPUTED and LOW-confidence skills move into the four-layer verification cascade. Those that pass are reinstated as MEDIUM or LOW, and those that fail are marked unverified and recommended for manual review or removal.

Algorithm 1 presents the complete consensus process, whereas Algorithm 2 details the steps of Phase 2 peer collaboration.

Algorithm 1: Multi-Agent Consensus Skill Extraction
Input: JD text D, agent set $P = \{p_1, p_2, p_3\}$
Output: skill set S_{final} with confidence labels
Phase 1 (Parallel independent extraction):
for $p \in P$ **do** $S_p \leftarrow$ Extract(p, D) **end for**
Phase 2 (Peer collaboration review):
$A \leftarrow \{S_p \mid p \in P\}$ // aggregate into shared context
for $p \in P$ **do** $S'_p \leftarrow$ Collaborate(p, D) **end for**
Phase 3 (Voting classification):
initialize counter dictionary $V \leftarrow \varnothing$
for $p \in P$ **do for** $s \in S'_p$ **do** $V[s.name].voters \leftarrow V[s.name].voters \cup \{p\}$ **end for end for**
for each $s \in V$: $r \leftarrow |s.voters| / |P|$**; if** $r = 1$ **then** $s.confidence \leftarrow$ **HIGH, else if** $r \geq 0.5$ **then MEDIUM, else LOW**
Phase 4 (External verification):
for each s with $s.confidence \in$ {**LOW**, **DISPUTED**}: VerifyCascade(s)
// KG → ESCO → Web → Manual
return $\{s \in V \mid s.confidence \neq$ **DISPUTED**$\}$

Algorithm 2: COLLABORATE (Peer Collaboration Review)
Input: agent p, original JD text D, peer result set $A = \{S_q \mid q \in P, q \neq p\}$
Output: corrected skill list S'_p
$S'_p \leftarrow S_p$ // initialize with current extraction result
for each $s_c \in \cup_{q \neq p} S_q$ **do**
if $s_c \in S_p$**: if** HasEvidence(s_c, D) **then** Keep(s_c) **else** Remove(s_c)
else: if HasEvidence(s_c, D) **then** $S'_p \leftarrow S'_p \cup \{$Add($s_c$)$\}$
end for
for each $s_p \in S_p$ not in $\cup_{q \neq p} S_q$ **do**
if ¬HasEvidence(s_p, D) **then** $S'_p \leftarrow S'_p \setminus \{s_p\}$
end for
for each group G **of synonyms do** $S'_p \leftarrow (S'_p \setminus G) \cup \{$Merge($G$)$\}$ **end for**
return S'_p

4.4 Consensus Metrics

To make inter-agent consistency measurable, we define the following metrics:

Inter-agent Jaccard similarity: J(p_i, p_j) = |S_{p_i} ∩ S_{p_j}| / |S_{p_i} ∪ S_{p_j}|, which gauges how much the skill sets of two agents overlap.

Diversity index: $\mathrm{Div}=1-\frac{1}{C_3^2}\sum_{i<j} J(p_i,p_j)$. High diversity indicates that the three models draw on different perspectives; when diversity falls too low, the multi-agent setup loses its point—the models simply emit near-identical results.

4.5 Four-Layer Verification Mechanism

Conditional trigger strategy. The guiding principle is "targeted verification": not every skill needs the full treatment. HIGH-confidence skills (those with unanimous votes from the three models) are trusted by default and skip verification; MEDIUM-confidence skills only fire a fast knowledge graph lookup; LOW and DISPUTED skills trigger the whole four-layer cascade, as drawn in Figure 3. The reason for this hierarchical design is chiefly economic: LLM calls and web searches cost latency and money, while knowledge graph lookup (L1) and ESCO alignment (L2) are local queries returning in milliseconds. Formally, the trigger condition is defined as:

verify(s) = { skip, if s.confidence = HIGH; {L1}, if s.confidence = MEDIUM; {L1, L2, L3, L4}, if s.confidence ∈ {LOW, DISPUTED} }

$$\text{verify}(s)=\begin{cases}\text{skip}, & \text{if } s.confidence=HIGH,\\ \{L1\}, & \text{if } s.confidence=MEDIUM,\\ \{L1,L2,L3,L4\}, & \text{if } s.confidence\in\{LOW,DISPUTED\}.\end{cases}$$

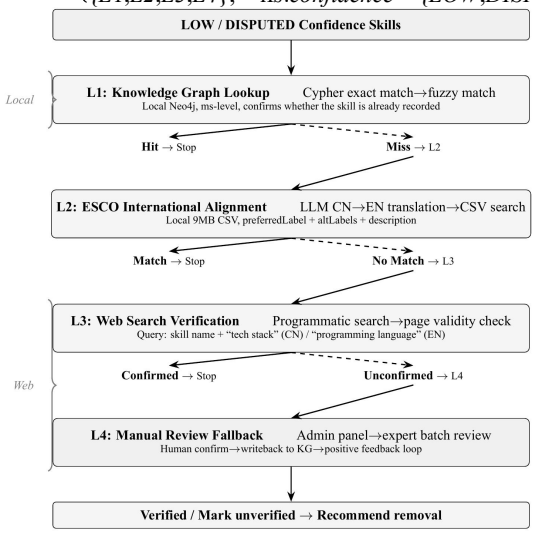

Figure 3. Four-Layer Progressive Verification Cascade (Terminates Upon First Confirmation)

4.6 L1: Knowledge Graph Lookup

The knowledge graph holds skill nodes that accumulate over many rounds of extraction and verification. Verification is carried out through Cypher queries: an exact match on the skill name returns the associated job count and a representative list of jobs; absent an exact match, fuzzy matching over the first four characters brings back similar skills for reference. This is the fastest and cheapest layer, but it can only tell us that a skill is known to the system—not that the skill genuinely exists in the world. Newly encountered skills that come up empty are not condemned as false on the spot; they wait for confirmation from the layers downstream.

4.7 L2: ESCO International Standard Alignment

ESCO—the European Skills, Competences, Qualifications, and Occupations classification—is a multilingual system maintained by the European Commission [18]; version 1.2.1 alone contains over 13,000 skill concepts reviewed by domain experts, which makes it a natural external reference for judging skill authenticity.

Marconi et al. [19] found that text-matching methods remain competitive in skill classification, lending support to our local CSV matching strategy. The core difficulty in this layer is the language gap between Chinese skills and English-language ESCO. We therefore use iFlytek Spark 4.0 Ultra, equipped with a translation cache, to render Chinese skills into English before matching them against ESCO's preferredLabel, altLabels, and description fields. The entire layer runs locally, incurring no network or API costs; its drawback is that ESCO is built around European industrial needs, so coverage of Chinese indigenous technology stacks is thin.

4.8 L3: Web Search Verification

Skills that escape both the knowledge graph and ESCO are taken to this layer, where search queries are constructed—appending "technology stack" for Chinese skills and "programming language OR framework" for English ones—and web searches are launched. If credible pages turn up on GitHub, technical documentation, or encyclopedic sites, the skill is treated as real; this is precisely how emerging and localized skills are verified.

4.9 L4: Manual Review Fallback

Should a skill still lack confirmation after the first three layers, it is flagged as unverified, with a recommendation to either review it manually or drop it from the final results. All such skills, together with their verification records, are surfaced in the admin console so that domain experts can review them in batches. Skills that humans eventually confirm are written back into the knowledge graph, where they serve as evidence for future automated verification, closing a positive feedback loop.

4.10 Knowledge Graph Construction and Evolution Tracking

4.10.1 Graph schema design

The job–skill knowledge graph is modeled as a property graph with two node types and a single relation type:

Position nodes carry attributes such as name, category, level, tech_stack, definition_json (core responsibilities, required/preferred skills, and application scenarios), source, and status.

Skill nodes carry skill_name, normalized_name, aliases, category/sub_category, esco_ref, trend_direction, and is_verified attributes.

REQUIRES relations run from Position →Skill, with attributes including tier (required/preferred/optional), importance_score, frequency, and source.

Graph construction is incremental. Each fresh extraction result is first attached to the existing graph by normalized-name matching; for nodes already present, relations are appended and frequencies are updated; and nodes that appear for the first time are marked to-be-verified (is_verified=0) until the four-layer verification process confirms them.

4.10.2 Dynamic evolution tracking

How job skill requirements shift over time is the central dynamic feature of our work. Each skill change event is logged in hr_skill_history. Whenever a fresh batch of JDs for some job produces skill sets that conflict with the historical record, the system diffs the two and emits three types of change events: added (new skills), removed (skills that disappeared), and modified (importance or level changes). Every record stores the observation date plus an evidence field pointing at the JD batch responsible; thus, the resulting evolution trends remain fully traceable.

From the accumulated history, the system computes each skill's annual/quarterly frequency and fits a trend slope using linear regression. A frequency that keeps rising with a significantly positive slope across two consecutive statistical periods marks the skill RISING; a negative slope marks it DECLINING; fluctuations within a threshold mark it STABLE; and a first-ever appearance marks it as NEW. The resulting trend direction and confidence are written into hr_skill_taxonomy, where they power front-end filtering and trend-based visualization.

4.10.3 Emerging job discovery

Beyond tracking how existing jobs evolve, the system also automatically uncovers emerging jobs from the JD corpus, working in tandem with the job discovery algorithm.

Candidate clustering: DBSCAN (ε=0.5 , min_samples=2) is run over JDs that existing job definitions do not yet cover, performing density-based clustering on skill vectors so that JDs with similar skill combinations are gathered into candidate job clusters.

Emerging job determination: A candidate cluster counts as an emerging job only if three conditions hold at once: (1) novelty of skill combination—the maximum Jaccard similarity against existing jobs in the knowledge graph

stays below 0.6; (2) sufficient demand—the cluster contains at least five JDs; (3) cross-enterprise appearance—at least two distinct companies are involved, excluding positions that stay inside a single enterprise.

Job definition generation: Once a cluster is validated, the iFlytek Spark LLM turns it into a structured job definition with five elements (job name, core responsibilities, required skills, preferred skills, and typical application scenarios); each definition is reviewed manually before being written into hr_positions and the knowledge graph.

From 3,942 JDs, the system automatically surfaced three emerging jobs—IoT+AI+RAG Engineer, Digital Twin/Metaverse Engineer, and AI Agent Engineer—all of which passed manual review.

4.10.4 Evolution tracking experiment results

Over a JD collection window of roughly 6 months, the system logged 37 skill change events: 19 added skills, 2 removed skills, and 16 importance or level changes. Table 2 presents the distribution of these evolution events by job category.

The AI field accounts for the most active skill churn (14/37), which squares with how fast its technology iterates. Among the added skills, "large model deployment," "vector database," and "AI agent development" track the 2024–2025 technology trends, while the removals "Hadoop" and "Struts" mirror the slow retirement of traditional technology stacks. Taken together, these observations confirm that the system can pick up real skill market signals from genuine JD data.

Table 2. Statistics of Job–Skill Evolution Events

Job category	Added	Removed	Modified	Total
Artificial Intelligence	8	0	6	14
Big Data	4	1	4	9
Cloud Computing	3	0	3	6
Information Security	2	1	2	5
IoT	1	0	1	2
Blockchain	1	0	0	1
Total	**19**	**2**	**16**	**37**

5. Experiments and Analysis

5.1 Dataset and Experimental Setup

We crawled our training data from BOSS Zhipin and Liepin between September 2024 and March 2025. Scraping turned out to be the hardest part of the whole project—both platforms deploy aggressive anti-crawling measures, and our first attempts were blocked almost immediately. We eventually switched to simulating a real browser, which got us through and let us collect the raw JDs we needed. After deduplication and cleaning, we kept 3,942 valid JDs covering six domains—artificial intelligence, big data, cloud computing, information security, IoT, and blockchain—gathered through 56 keyword groups. A random sample of 200 JDs was then annotated by three of us using four entity types (SKILL, KNOWLEDGE, TOOL, and SOFT), and the inter-annotator agreement (Fleiss' $\kappa = 0.76$, $p < 0.001$) gave us confidence in the labels. Detailed statistics of the dataset appear in Table 3

Table 3. Statistics of the Experimental Dataset

Domain	JD count	Skills/JD	Rep. jobs
Artificial Intelligence	1247	8.3	LLM engineer, algorithm researcher
Big Data	918	7.0	Data development engineer, ETL engineer
Cloud Computing	783	7.5	DevOps engineer, cloud architect
Information Security	512	6.2	Penetration tester, security operations
IoT	296	5.8	Embedded development, IoT architect
Blockchain	186	5.3	Smart contract development, Web3 engineer
Total	**3942**	**7.2**	—

Every experiment was run on identical hardware: an Intel Xeon Gold 6338 CPU @ 2.0GHz, 128GB of RAM, and an NVIDIA A100 40GB GPU. We evaluate with precision (P), recall (R), and F1-score, taking manual annotation as the gold standard; on the efficiency side, we record the average processing time per JD and end-to-end throughput.

5.2 Skill Extraction Comparison and Statistical Analysis

To determine whether the multi-agent consensus mechanism helps, we designed the following comparison schemes.

Baseline 1 (traditional NER): BiLSTM-CRF model [5] trained on labeled data.

Baseline 2 (single best LLM): the single model that attains the highest F1 among the three LLMs.

Baseline 3 (single-model three-sample voting):

one LLM sampled three times at different temperatures (T=0.0,0.2,0.4) and then combined by majority voting.

Proposed method (three-model consensus): full four-phase consensus mechanism (Phases 1–4).

Proposed method (ablation): Phase 2 peer collaboration is removed, leaving only the parallel extraction and voting.

The experimental results are presented in Table 4

Table 4. Results of the Skill Extraction Comparison Experiments

Method	P(%)	R(%)	F1(%)
BiLSTM-CRF (Baseline 1)	78.4	72.8	75.5
Single best LLM (Baseline 2)	86.2	83.5	84.8
Single-model three-sample voting (Baseline 3)	87.0	84.1	85.5
Proposed (ablation, no collaboration)	88.3	85.6	86.9
Proposed (full)	**90.5**	**87.6**	**89.0**

To convince ourselves that these gains are not due to chance, we ran McNemar tests on 200 labeled samples. The proposed method differs significantly from the single best LLM (p<0.01, 95% CI [1.8,6.6]), the ablation version (p<0.05, 95% CI [0.5,3.7]), and BiLSTM-CRF (p<0.001, 95% CI [8.2,18.8])—strong evidence that the gains did not arise from random fluctuations.

5.3 Computational Cost Analysis

Table 5 lays out the computational cost of processing 3,942 JDs. Each JD calls for 6 LLM calls (3 parallel calls in Phase 1 plus 3 parallel calls in Phase 2). With three-thread parallel processing, one JD takes 18 seconds, the whole corpus finishes in 6.1 hours, and the total API bill is 276 CNY. By comparison, manual annotation-based verification at 50 CNY per JD would set us back 197,100 CNY at this scale. In short, the proposed method brings the cost of extraction and verification down to roughly 0.14% of the manual approach.

Table 5. Computational Cost of Processing 3,942 Jds

Indicator	Value
LLM calls per JD	6 (3 in Phase 1 + 3 in Phase 2)
Processing time per JD (single thread)	18 s
Full processing time (3 threads)	6.1 h
Full processing time (single thread)	18.3 h
API cost per JD (three models)	0.07 CNY
of which: iFlytek Spark (2 calls)	0.04 CNY
of which: DeepSeek (2 calls)	0.02 CNY
of which: GLM-4 (2 calls)	0.01 CNY
Total API cost for 3,942 JDs	276 CNY
Manual annotation comparison (50 CNY/JD)	197,100 CNY

5.4 Result Analysis and Ablation Studies

Table 4 presents two mechanisms at work. The first is the complementarity of cross-vendor models: F1 rises from 84.8% (single best LLM) to 89.0% (full method), a clear sign that different vendors' models hold complementary strengths where one another's knowledge is blind. The second is the concrete contribution of peer collaboration: the ablation version (voting only) already improves over the single best LLM by 2.1 percentage points (84.8%→86.9%), and the full method gains another 2.1 points (86.9%→89.0%)—so Phase 2 earns its keep on its own. By contrast, single-model three-sample voting (Baseline 3) improves by a mere 0.7 points (85.5%), confirming once more that resampling the same model cannot yield real diversity gains.

To answer the question "why three agents," we tested the consensus effect with different numbers of agents (K=2,3,4) on 200 labeled samples. K=2 keeps only Spark (strict) and DeepSeek (balanced); K=3 is the full configuration of this paper; K=4 adds Baichuan 3.0 (explorer, T=0.4). Phase 2 collaboration is enabled in every configuration. The results appear in Table 6

Table 6. Agent Number Ablation Experiment

Config	P(%)	R(%)	F1(%)	Div	Time (s/JD)	Cost (CNY)
K=2 (Spark+DeepSeek)	88.6	86.0	87.2	0.22	12	0.05
K=3 (proposed full)	**90.5**	**87.6**	**89.0**	0.18	18	0.07
K=4 (+Baichuan 3.0)	90.8	87.8	89.2	0.16	24	0.09

We expected more agents to help, but the numbers told a different story: moving from two to three agents yields a sizeable gain (+1.8 points), whereas a fourth agent adds only 0.2 points, while time and cost climb by 33% and 29%, respectively. Three complementary cross-vendor models already cover the main cognitive blind spots; a fourth model contributes little at the margin—consistent with the pilot, where Baichuan never cracked the top three.

Phase 2 currently performs a single round of collaboration. To determine whether iterative

multi-round debates pay off, we extended Phase 2 to two and three rounds (in every round, agents may consult the outputs of all peers from the round before). Table 7 reports the results for 200 labeled samples.

Table 7. Collaboration Round Ablation Experiment

Rounds	P(%)	R(%)	F1(%)	Div
0 (voting only, no collaboration)	88.3	85.6	86.9	0.40
1 (proposed default)	90.5	87.6	89.0	0.18
2	90.7	87.8	89.2	0.11
3	90.7	87.7	89.1	0.07

The biggest jump occurs between 0 and 1 round (+2.1 points), underlining the core value of collaboration; 2 rounds edge out 1 round by only 0.2 points (89.0% → 89.2%), and 3 rounds are essentially flat (89.1%), while diversity keeps sliding (from 0.18 to 0.07) as agents drift toward output homogenization. This mirrors the sycophancy effect that CONSENSAGENT [17] describes: endless debate pushes agents toward convergence rather than toward more valuable dissensus. For this task, then, a single round of collaboration is the sweet spot.

Our central claim is that cross-vendor model diversity beats within-model temperature diversity. We therefore compared two configurations: (A) same-model temperature diversity, in which Spark alone is sampled three times at T=0.0,0.2,0.4 to impersonate three agents (Baseline 3); and (B) cross-vendor, taking one instance each of Spark, DeepSeek, and GLM-4 (the proposed method). Both ran the full four-phase consensus, and the results on 200 labeled samples are in Table 8

Table 8. Same-Model Temperature Diversity vs. Cross-Vendor Model Diversity

Config	P(%)	R(%)	F1(%)	Div	FP
Same-model temperature (Spark×3)	87.0	84.1	85.5	0.15	7.8
Cross-vendor (Spark+DeepSeek+GLM-4)	**90.5**	**87.6**	**89.0**	**0.31**	**3.1**

This result directly confirms the design choice we made in Section 4.2: cross-vendor wins by 3.5 F1 points over same-model temperature diversity (85.5%→89.0%), doubles the diversity from 0.15 to 0.31, and cuts the false-positive rate from 7.8% to 3.1%. The underlying reason is straightforward: varying the temperature of one model only stirs random perturbation inside the same sampling distribution, so outputs remain highly correlated; in contrast, models from different vendors carry systematic differences in pretraining data, alignment strategies, and knowledge boundaries, display complementary strengths and blind spots across different parts of the same JD, and let the voting mechanism fuse that complementary information effectively. This ablation is, in effect, direct empirical support for choosing cross-vendor models over multiple instances of one model.

This result also directly supports our core claim: diversity rooted in systematically different pre-training corpora and alignment strategies fundamentally beats within-model temperature sampling, which can only nudge the same decision boundary.

5.5 Verification and Error Analysis

To assess the effectiveness of the four-layer verification mechanism, we randomly sampled 200 skills from the LOW and DISPUTED confidence levels, judged their authenticity (genuine vs. fabricated) by hand, and compared these judgments with the verification results of the system. Table 9 presents the hit rate and accuracy of each layer.

Table 9. Effectiveness of Each Verification Layer

Layer	Hit rate (%)	Accuracy (%)	Time (ms)
L1: Knowledge graph lookup	45.2	93.5	12
L2: ESCO alignment	32.8	88.7	45
L3: Web search	15.6	81.2	1,240
L4: Manual review	6.4	100.0 (reference)	—

A cascade effect is clearly visible: because verification "terminates upon first confirmation," the volume of work shrinks layer by layer. L1 takes on the largest share of verification requests as it is first; L2 picks up internationally recognized skills that the graph failed to confirm; L3 deals mainly with localized and emerging skills—few in number but highest in verification value, since the two layers before it cannot handle them; and L4 steps in only when all three earlier layers come up empty.

ESCO alignment rate here refers to the proportion of real skills among the 200 labeled samples that were successfully matched in ESCO—in other words, how well the extraction results line up with the international standard. Because ESCO is overwhelmingly in English, how well Chinese skills are translated directly shapes this rate. Table 10 reports the alignment results under different translation strategies.

LLM-based translation raises the ESCO

alignment rate from 18.7% to 71.5% (+52.8 pp); adding a cache then cuts the per-skill time from 1,180 ms to 340 ms (a 71.2% reduction). Larger batches also help: cache hit rates increase, and the average time decreases.

Table 10. ESCO Alignment Rate under Different Translation Strategies

Strategy	Alignment (%)	Time (ms)
No translation (direct Chinese matching)	18.7	0.4
LLM translation (no cache)	67.3	1,180
LLM translation (with cache)	71.5	340

Case 1 (divergence convergence): in an operations engineer JD, the curious-explorer agent went ahead and annotated Kubernetes and Docker even though the original text merely said "familiar with containerization technology." The collaborative review retained these annotations, reasoning that "containerization technology is equivalent to Docker + Kubernetes"; all five skills ended up passing at MEDIUM/HIGH.

Case 2 (hallucination suppression): in a full-stack engineer JD, the curious-explorer agent over-inferred and annotated GraphQL, gRPC, and other technologies absent from the original text. The strict and balanced agents rejected these annotations during collaborative review—"no evidence in the original text"—and the voting round dropped them. Had a single high-temperature LLM been in charge, those false skills would have flowed straight into the knowledge graph.

To examine the limitations of the proposed method, Table 11 breaks down the precision, recall, and F1-score of the full method by entity type.

Table 11. Error Analysis by Entity Type (Proposed Full Method)

Entity type	P(%)	R(%)	F1(%)	Sample (%)
SKILL (concrete technology)	92.1	89.3	90.7	48.2
TOOL (tool/platform)	93.6	91.5	92.5	31.5
KNOWLEDGE (theoretical system)	86.8	82.4	84.5	14.8
SOFT (soft skill)	81.2	78.6	79.9	5.5
Overall	**90.5**	**87.6**	**89.0**	100.0

The method does best on SKILL and TOOL (F1 above 90%), not surprisingly, since these entities come with clear technical names and relatively consistent boundary judgments. KNOWLEDGE trails at 84.5%, weighed down by the fuzzy line between theoretical systems and specific skills; SOFT lands lowest at 79.9%, because soft skills are expressed in wildly varied ways across Chinese JDs. Since SKILL and TOOL together make up about 80% of the samples, they effectively dictate the overall F1-score.

To determine which verification layer earns its keep, we stripped the layers one at a time, counted how many false skills were then wrongly kept, and recorded the resulting F1 changes on 200 labeled samples. Table 12 summarizes the results.

Table 12. Four-Layer Verification Ablation Results

Setup	False skills	F1(%)	ΔF1 gap (pp)
No verification (consensus only)	31	84.7	−4.3
KG only (L1)	18	87.2	−1.8
KG + ESCO (L1+L2)	9	88.4	−0.6
KG + ESCO + Web (L1+L2+L3)	6	88.8	−0.2
Full four layers (L1+L2+L3+L4)	**2**	**89.0**	—

The ablation results read as follows: (1) L1 on its own filters 41.9% of the false skills ($31 \rightarrow 18$); (2) L2 supplies the biggest increment ($18 \rightarrow 9$), underlining the unique value of ESCO alignment; (3) L3 trims false skills down to 6, catching emerging skills that ESCO does not cover; (4) L4 eliminates the last 4. With consensus alone, F1 sits at 84.7%; the full four layers raise it to 89.0% (+4.3 pp)—a gain on par with what multi-agent consensus achieves over a single LLM (+4.2 pp).

5.6 System Evaluation

The inter-agent Jaccard similarity and diversity index defined in Section 4.4 provide quantitative backing to the interpretability of the consensus mechanism. On 200 labeled samples, the pairwise Jaccard similarities immediately after Phase 1 independent extraction are: Spark vs. DeepSeek $J=0.62$, Spark vs. GLM-4 $J=0.57$, and DeepSeek vs. GLM-4 $J=0.60$, with a diversity index of $Div=0.40$. Once Phase 2 collaboration begins, the diversity falls to 0.18 as the three models converge—an expected consequence, since agents temper their extreme judgments after seeing peers' results.

When the models diverge, three causes dominate: inconsistent skill granularity (42%), ambiguous JD expressions (35%), and differing domain knowledge boundaries (23%).

Across the graph built from all 3,942 JDs, the false-skill rate hovers around 15.0% without

verification and falls to 1.4% once the full four-layer verification is applied—a 90.7% reduction— whereas 25.3% of skill nodes are deduplicated.

For the end-to-end evaluation, we assembled 120 real JDs, collected 80 resume samples from volunteer students, and built 60 job–person matching pairs that three of us annotated, reaching a Fleiss' κ of 0.79, as shown in Table 13.

Table 13. Results of the end-To-End Key Metric Evaluation

Metric	Test scale	Accuracy (%)	Threshold
JD parsing accuracy	120 JDs	**95.3**	≥90
Resume extraction accuracy	80 resumes	**93.7**	≥90
Job–person matching accuracy	60 matching pairs	**92.4**	≥90

All three metrics clear the 90% acceptance bar. JD parsing leads at 95.3%, thanks to the gains from multi-agent consensus in skill extraction and the fallback filtering of the four-layer verification. Resume extraction sits at 93.7%—capped by the variety of resume layouts and the quality of PDF scans, though it tops 96% on standard templates. Job–person matching reaches 92.4% by combining skill coverage, experience level, and semantic reasoning over the job–skill graph, beating pure keyword matching (about 78%) by a wide margin.

Beyond the experiments, the system supports a broad set of downstream applications in talent management and career services. Enterprises can draw on the job–skill knowledge graph for skill gap analysis, drafting recruitment requirements, and workforce planning. Individuals get personalized learning path recommendations and career development advice built on current and predicted skill trends. Educators and policymakers, meanwhile, can treat the dynamic evolution tracking capability as empirical evidence for curriculum design and vocational training, which helps close the gap between education and the labor market.

6. Conclusion

This paper set out to remove two bottlenecks in job–skill knowledge graph construction—unreliable skill extraction and the absence of automated verification—by combining multi-agent consensus with four-layer verification, and by building an end-to-end system that spans data collection, skill extraction and verification, graph storage, and visualization. The experiments bear this out: the proposed method beats single models and traditional NER baselines on both precision and recall, the four-layer verification markedly lowers the chance that hallucinated skills enter the knowledge graph, and all three end-to-end metrics (JD parsing, resume extraction, and job–person matching accuracy) exceed the 90% acceptance threshold.

Several directions remain open. We intend to (1) introduce multi-round iterative debate guided by anti-sycophancy protocols [17]; (2) map the taxonomy onto the Occupational Classification Code of the People's Republic of China (2022 edition) so the system aligns with domestic occupational standards; (3) put in place a long-term data collection pipeline to follow 3–5 year skill lifecycles; and (4) deploy and validate the system on partner enterprise recruitment platforms, folding review feedback back into the incremental fine-tuning loop.

Limitations. The results are encouraging, but several limitations temper them. First, ESCO is built around European industrial needs, so it covers Chinese indigenous technology stacks—HarmonyOS, Kylin OS, and the like—only sparsely, and localized skills therefore align at a relatively low rate. Second, the experimental data come from a handful of recruitment platforms observed over a short window, so the full dynamics of the labor market may not be captured. Third, evaluation annotation was done by graduate students rather than industry experts, which introduces some subjectivity into the gold standard.

References

[1] He T, Zhang Q, Zheng G, et al. A Survey of Knowledge Graph Construction in the Era of Large Language Models. Control and Decision, 2025, 40(12): 3509–3527. (in Chinese)

[2] Yang Y W, Zhai P Y, Jin Y. Job Entity Recognition and Talent Profiling in the Digital Industry Based on BERT and BiGRU. Journal of Wuhan University (Natural Science Edition), 2025(5): 1–12. (in Chinese)

[3] Wang Y S, Shi Y W, et al. Competence Graph of "Gold Majors" in Vocational Education: Logic, Construction Mechanism, and Practice Path. Chinese Vocational and

Technical Education, 2024(32): 15–23. (in Chinese)
[4] Zhang J, Huang W F, Wu C J, et al. A Survey of LLM-Enhanced Knowledge Graph Construction and Reasoning. Computer Science and Exploration, 2025, 19(11): 2855–2872. (in Chinese)
[5] Wang Z Q, Song J X, Yu S S, et al. Construction Method of Smart Mine Knowledge Graph Based on Dependency Parsing. Mining Research and Development, 2023, 43(10): 180–186. (in Chinese)
[6] Zhang B A, Cao R Q, et al. Design and Implementation of Vertical Domain Knowledge Graph Construction and Application Platform. Frontiers of Data and Computing, 2023, 5(3): 78–91. (in Chinese)
[7] Zhu Y, Wang X, Chen J, et al. LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities. World Wide Web, 2024, 27(5): 58.
[8] Yu J L, Su Z W, et al. Employee Training Recommendation Based on Knowledge Graph and Collaborative Filtering. Journal of Yunnan Minzu University (Natural Sciences Edition), 2025(5): 1–9. (in Chinese)
[9] Wang Y J, Pan H P. Human Resource Job Matching Prediction Based on Knowledge Graph and Attention Mechanism. Journal of Guiyang University (Natural Sciences Edition), 2025(4): 45–50. (in Chinese)
[10] Chen J, Saha S, Bansal M. ReConcile: Round-Table Conference Improves Reasoning via Consensus among Diverse LLMs//Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL). Bangkok, 2024: 7066–7085.
[11] Zhang Y, Sun R, Chen Y, et al. Chain of Agents: Large Language Models Collaborating on Long-Context Tasks//Advances in Neural Information Processing Systems (NeurIPS). 2024, 37.
[12] Lu M, Xie Y, Bi Z, et al. CROSSAGENTIE: Cross-Type and Cross-Task Multi-Agent LLM Collaboration for Zero-Shot Information Extraction//Findings of the Association for Computational Linguistics (ACL). Vienna, 2025: 13953–13977.
[13] Haji F, Bethany M, Chiang C, et al. Reflective Agreement: Combining Self-Mixture of Agents with a Sequence Tagger for Robust Event Extraction//Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). Suzhou, 2025: 30476–30492.
[14] Ye H, Gui H, Zhang A, et al. CooperKGC: Beyond Isolation—Multi-Agent Synergy for Improving Knowledge Graph Construction//Proceedings of the China Conference on Knowledge Graph and Semantic Computing (CCKS). 2024. arXiv:2312.03022.
[15] Kobayashi T, Nakatsuji M. Towards Adaptive Knowledge Structuring by Multi-Agent Consensus//CEUR Workshop Proceedings. 2024, 4093: 1–12.
[16] Pan S, Luo L, Wang Y, et al. Unifying Large Language Models and Knowledge Graphs: A Roadmap. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(7): 3580–3599.
[17] Pitre P, Ramakrishnan N, Wang X. CONSENSAGENT: Towards Efficient and Effective Consensus in Multi-Agent LLM Interactions Through Sycophancy Mitigation//Findings of the Association for Computational Linguistics (ACL). Vienna, 2025: 22112–22133.
[18] de Smedt J, le Vrang M, Papantoniou A. ESCO: Towards a Semantic Web for the European Labour Market//Proceedings of the Workshop on Linked Data on the Web (LDOW 2015), CEUR Workshop Proceedings, Vol. 1409. Florence, Italy, 2015.
[19] Marconi G, Baer I, Da Silveira M, et al. Evaluating ESCO Skill Classifiers. Statistical Journal of the IAOS, 2025, 41(4): 1038–1057.

