LMABC: A Cost-Efficient Large Language Model Multi-Agent Framework for Automated Sequence Labeling in Building Codes

Xingchen Hu*, Zhenjun Ma†, Binbin Yong‡, Jun Shen*
*School of Computing and Information Technology, University of Wollongong, Australia
†Sustainable Buildings Research Centre, University of Wollongong, Australia
‡School of Information Science and Engineering, Lanzhou University, China
*xh739@uowmail.edu.au; jshen@uow.edu.au
†zhenjun@uow.edu.au
‡yongbb@lzu.edu.cn

***Abstract*—Automated sequence labeling of regulatory documents is crucial for automated compliance checking (ACC) and domain knowledge graph (KG) creation, yet is hindered by a "high-cost, low-resource" dilemma. Public, high-quality labeled datasets for building codes are scarce and difficult to reuse across regions, making large-scale annotation expensive. Although large language models (LLMs) show strong few-shot capabilities, multi-round, clause-by-clause annotation with high-performance APIs remains prohibitively costly, whereas affordable models are less reliable. To resolve the cost-performance dilemma, we propose LMABC, a low-cost, low-resource LLM Multi-Agent framework for Automated Sequence Labeling in the Building Codes. Rather than relying on a single powerful model, LMABC coordinates multiple diverse, affordable LLM agents to achieve high-quality annotation through collective intelligence. The framework couples a task allocation mechanism with a two-phase labeling workflow consisting of parallel labeling and chain-based transfer labeling. Final results are obtained by a hybrid consensus ensemble that combines majority voting with dynamic weighting to support downstream KG creation. Experiments on building-code and general-domain benchmarks demonstrate that LMABC approaches high-performance models at a fraction of the cost, and achieves higher accuracy with fewer tokens than state-of-the-art (SOTA) methods on general tasks.**

***Index Terms*—Automated Compliance Checking, Automated Sequence Labeling, Building Codes, Large Language Model, Multi-Agent Chain**

I. Introduction

Strictly compliance is essential for safety and efficiency in the architecture, engineering, and construction (AEC) industry. Traditional manual compliance checking of building represented by computer-aided design (CAD) drawings and building information modelling (BIM) models clause-by-clause is inefficient, expensive, and error-prone [1]. Consequently, automated compliance checking (ACC), also known as automated rule checking (ARC) has become increasingly adopted [2]. A key bottleneck is translating unstructured regulatory clauses into machine-interpretable knowledge, which typically requires large, high-quality annotated corpora [3], [4].

From hard-coded rules to pretrained deep learning models, ACC has consistently relied on large high-quality pre-annotated corpus for knowledge graph (KG) creation, However, regional differences limit data transferability, creating a "high-cost, low-resource" dilemma in the AEC industry [5]. Although large language models (LLMs) offer the zero/few-shot potential [6], the cost-reliability trade-off of single models has driven recent research towards collaborative multi-agent frameworks using affordable LLMs [7]. Although consensus can improve robustness, repeatedly labeling the same tasks consumes substantial tokens, and few-shot methods still require an initial annotated set. As a result, existing solutions remain costly and prone to unstable consensus [8], [9]. In short, ACC remains bottlenecked by the lack of high-quality annotated data, and even LLM-based text annotation still needs a seed dataset. When a single high-performance model is prohibitively expensive (via either API or on-prem deployment), an important challenge is to coordinate multiple affordable LLMs to reduce token use while improving accuracy.

Motivated by this insight, we propose **LMABC**, a low-cost, low-resource **L**LM **M**ulti-Agent framework for **A**utomated Sequence Labeling **B**uilding **C**odes. LMABC supports downstream domain KG creation and ACC by integrating data preprocessing, task allocation, a multi-agent labeling workflow, and a consensus ensemble. Guided by ACC requirements, we normalize text and construct an ontology to support pre-annotation and guide sequence labeling [10]. Task allocation balances throughput and re-annotation cost, while the two-phase workflow combines parallel labeling with chain-based transfer labeling so that complex tasks can be re-annotated by stronger agents. Meanwhile, prompt optimization further reduces token consumption [11]. Final results are produced by a hybrid consensus method that combines majority voting with dynamic weighting based on standardized accuracy, relative rank, and perplexity-based evidence.

LMABC enables multiple affordable agents to achieve high-quality regulatory sequence labeling at a fraction of the cost of a single state-of-the-art (SOTA) model. Across domain and general benchmarks, it achieves competitive accuracy and offers an extensible corpus pipeline for KG-based ACC task.

II. Related Work

A. ACC in Building Construction Standards

Early ACC methods relied on hard-coded rules, which lacked flexibility because updates required manual recoding. They also could not process unstructured natural language, a major limitation given the complexity of regulations [12], [13].

Advances in machine learning (ML) and deep learning (DL) methods for natural language processing (NLP) shifted ACC toward transforming unstructured regulations into machine-interpretable representations [14]. Traditional ML methods rely on handcrafted features for text classification and information extraction [15], whereas DL methods learn semantic features automatically in an end-to-end manner to extract entities and relations from regulatory documents and organize them into ontology, executable rule procedures or KG [16], [17], improving the extraction of complex compliance rules.

More recently, generative AI (GenAI) represented by LLMs, enables the direct rules extraction from regulations by zero-shot or few-shot learning capabilities, further reducing the reliance on annotated data [18], [19]. Nevertheless, ACC research relies heavily on structured building data (CAD, BIM) and unstructured regulatory text data, necessitating high-quality annotated corpora for robust rule development and validation. Even with advancements in pre-trained models, high-quality data is still essential for the best accuracy [20].

Hence, acquiring high-quality expert-annotated corpora is costly and labor-intensive, underscoring the critical need for cost-effective, efficient, and automated annotation solutions.

B. Automated Text Annotation Methods

Text annotation assigns meaningful labels to raw text and is a foundational prerequisite for training supervised learning models in nearly all NLP tasks [21]. As indicated in Section II-A, ACC often emphasizes sequence labeling data and some POS tagging data to support ontology and KG construction.

Early automated annotation methods relied on rule-based and dictionary-based systems that used handcrafted patterns to label text, which is interpretable but hard to generalize and costly to maintain [22]. Subsequently, similar as ACC methods, rule-based approaches were followed by ML and DL based methods, however, they typically require substantial training data and computing resources and might degrade on domain-specific texts without proper adaption [23], [24].

Recently, LLMs have enabled zero-shot or few-shot annotation, effectively recognize entities and relationships without task-specific training [25]. However, a single LLM can hallucinate or be overconfident in specialized domain settings, and high-performance models are expensive via either on-premises deployment or API access [26]. This motivates multiple LLMs collaborate together as a multi-agent system to improve reliability under constrained budgets.

C. LLMs-based Multi-Agent System

By utilizing each LLMs as cognitive core, multi-agent systems boost the reasoning and planning abilities of autonomous agents, enabling efficient collaboration on complex tasks [27].

For text generation and annotation, a common paradigm is multiple agents solve the same task in parallel and then merge outputs by consensus, improving reliability but substantially increasing token and costs [28]. Meanwhile, consensus is typically achieved from majority voting, LLM-as-Judge, multi-agent debate, or manual judgment, which needs extra rounds or external judges thus increase the cost [29], [30]. Some novel schemes such as dynamic weighting [31] and relative ranking [32], which directly reduce extra LLM calls by directly scoring or comparing agent outputs. However, these methods can still fail due to the overconfidence of a single LLM.

III. Method

In this section, we introduce LMABC, a cost-efficient LLM multi-agent framework for automated sequence labeling in building codes, the framewoke is illustrated in Fig. 1.

A. Data Preprocessing

To build an annotation dataset for ACC task in the Australian AEC industry, LMABC uses publicly available Australian regulatory, law, regulation documents as primary raw sources, including the National Construction Code (NCC) and Australian/New Zealand Standard (AS/NZS) series, etc. These documents are publicly available online in PDF format.

To improve labeling efficiency, we preprocess thethe raw documents before sequence labeling. We also construct a lightweight domain ontology by adapting Qiu's method [33] and integrating resources such as buildingSMART Data Dictionary (bSDD) and Brick. Tailored to Australian building regulatory text, the ontology focuses on building energy and structural elements and supports sequence labeling, named entity recognition (NER), KG creation, and ACC.

B. Multi-Agent Labeling Workflow

1) Few shot Prompting: Prompt design is the first step in our multi-agent labeling workflow. Since zero-shot falter on complex, domain-specific tasks and fine-tuning model is infeasible due to its need for large annotated datasets, which is a direct conflict with our low-resource, low-cost scenario, we adopt the few-shot prompting strategy [34]. To better exploit the natural language processing capabilities of LLMs while avoiding mismatches between extracted entities and assigned labels, we enforce a structured natural language output, each agent produces "Label 'A' as 'B' " statements to ensure a one-to-one mapping between text spans and labels.

2) Multi-Agent Labeling: After setting up the prompt, we herein present the multi-agent sequence labeling workflow of LMABC, which can be divided into two phases. The first phase involves parallel labeling, while the second phase employs chain-based transfer labeling. Therefore, LMABC can be represented as Φ, the whole workflow can be denoted as:

$$\Phi = \left(\|_{i=1}^{n} L_{i,\mathrm{p1}}\right) \rightarrow \left(L_{1,\mathrm{p2}} \Rightarrow L_{2,\mathrm{p2}} \Rightarrow \cdots \Rightarrow L_{n,\mathrm{p2}}\right)$$

Among them, $\left(\|_{i=1}^{n} L_{i,\mathrm{p1}}\right)$ denotes the parallel labeling in Phase 1. $(L_{1,\mathrm{p2}} \Rightarrow L_{2,\mathrm{p2}} \Rightarrow \cdots \Rightarrow L_{n,\mathrm{p2}})$ denotes the chain-based sequential transfer labeling in Phase 2.

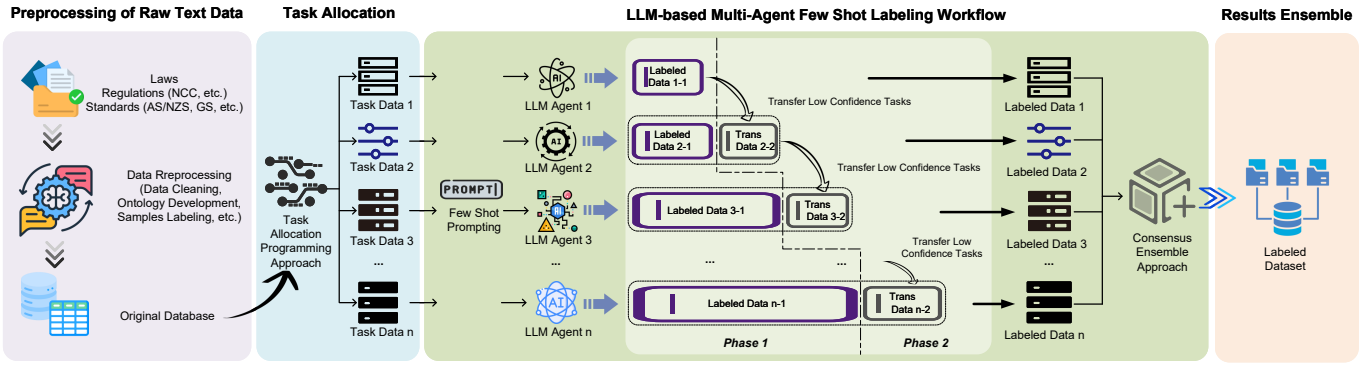

Fig. 1. Overview of LMABC. Data Preprocessing (Section III-A) prepares inputs for sequence labeling. Task Allocation (Section III-C) determinesagent order and assigns initial tasks. The Multi-Agent Labeling Workflow (Section III-B) consists of two phases: Phase 1 parallel labeling and Phase 2 chain-based transfer labeling. Consensus Ensemble (Section III-D) generates the final result for each task.

a) Phase 1: Parallel Labeling: As the initialization for Phase 1, the total task dataset D is partitioned into n subsets $\{D_1, D_2, \cdots, D_n\}$. Each LLM agent L_i is allocated a subset D_i as its initial set of labeling task. Subsequently, all agents work in parallel to concurrently annotate all their tasks.

b) Phase 2: Chain-based Sequential Transfer Labeling: Phase 2 adopts a chain framework, means the commencement of this phase for any agent L_i is strictly contingent upon the completion from the preceding L_{i-1}. The goal of Phase 2 is to improve final accuracy by subjecting a subset of tasks to re-annotate by other agents, which provides the foundation for a consensus ensemble. Accordingly, each agent follows a "receive, re-annotation, and transfer" principle.

First, L_i receives and re-annotates the task set $TD_{i-1,p2}$ transferred from its upstream L_{i-1}. Then L_i consolidates the re-annotated tasks with its own Phase 1 tasks D_i into a complete dataset $D_{i,p2}$, and then transfers a selected subset $TD_{i,p2}$ to the next L_{i+1} for its Phase 2 work.

In Phase 2, task transfer between agents is governed a confidence mechanism based on LLM output probabilities and a dynamic transfer ratio. In this study, we use perplexity (PPL) to measure the confidence of each task output. PPL is calculated from the token-level log probabilities (logprobs) returned by the LLM during generation. As shown in Eq. (1), a higher PPL indicates greater uncertainty and lower confidence.

$$\mathrm{PPL}(Res) = \exp\Big(-\tfrac{1}{N_{Res}} \textstyle\sum_{i=1}^{N_{Res}} \log P(Res_i \mid Res_1, \ldots, Res_{i-1})\Big) \quad (1)$$

where, Res is the output token sequence. N_{Res} is the total number of tokens in Res. The agent L_i transfers a fraction $Ratio_i = (1 - D_i/D)$ of task to its downstream L_{i+1}. Hence, L_i ranks all tasks in its consolidated dataset by their PPL in ascending and transfers top $Ratio_i$ high-PPL tasks to L_{i+1}.

C. Task Allocation

Agents in LMABC differ in performance such as accuracy and throughput, making effective initial task allocation critical to overall system performance. We design a task allocation mechanism to balance two key goals. First, it keeps the workflow synchronized, allowing downstream agents to start Phase 2 as soon as upstream agents forward low-confidence tasks. Secondly, it maximizes the number of transferred tasks, which improves the reliability of the final results by providing stronger support for the consensus ensemble. Therefore, initial task allocation planning is a core component of our framework. Accordingly, LMABC uses a two-part allocation scheme comprising an initial task constraint and agent sequencing rules.

To improve the final accuracy, we place the faster with higher accuracy agents as downstream agents in the chain framework as safety nets. However, processing speed of a LLM is primarily constrained by TPM (tokens per minute) and RPM (requests per minute), Besides, when multiple LLMs run in parallel, they can further degrade under parallel execution due to shared-quota contention. Therefore, we define the capacity coefficient K_i for agent L_i to measures the maximum number of tasks per minute that can stably process, RPM_i and TPM_i are PRM and TRM of L_i, as shown in (2).

$$K_i = \min\left\{\mathrm{RPM}_i, \frac{\mathrm{TPM}_i}{T_{avg}}, p_i' \cdot R_{total}\right\} \quad (2)$$

where, T_{avg} is the average tokens per task. p_i' is the benchmark parallel throughput ratio of L_i (its workload share under concurrency). R_{total} is the total throughput of the multi-agent observed during the parallel test (tasks per minute).

We then order agents in the chain using a bubble-sort-like rule based on K_i and historical accuracy $Acc(L_i)$. The pseudo-code is detailed in Algorithm 1.

For initial task constraint, although pursued synchronization is an ideal objective, we use linear constraints to allocate initial task to approach this state as closely as possible.

D. Consensus Ensemble

Upon the completion of the Phase 1 and 2 processes, many tasks will have received repeated labeling results from different agents. This intentionally designed "redundancy" in labeling provides the necessary foundation for a final consensus step. In this study, we proposed a consensus ensemble mechanism to "arbitrate" these differing output results, fuse conflicting labels, and yield a single, definitive result for each task.

Our consensus ensemble mechanism combines two common consensus strategies in multiple LLMs: majority voting and dynamic weighted voting. It considers three key aspects: the agent's overall historical accuracy in annotating all the task by

Algorithm 1 Chain Sequence with Accuracy-First and Capacity-Aware Backward Pass

InitializeSequence($\mathcal{L}$, acc, K):

Inputs: agent set $\mathcal{L} = \{L_1, \ldots, L_n\}$; accuracy $\mathrm{acc}(L_i)$; capacity K_i

Thresholds: $\tau_{\mathrm{acc}} \leftarrow 0.05$; $\rho \leftarrow 1.5$

Baseline sort: $S \leftarrow \mathrm{sort}(\mathcal{L})$ by $\mathrm{acc}(L_i)$ in ascending order

AdjustSequence((S)

for $i \leftarrow |S| - 2$ **down to** 0 **do**

  $L_{\mathrm{up}} \leftarrow S[i]$

  $L_{\mathrm{down}} \leftarrow S[i+1]$

  $\Delta_{\mathrm{acc}} \leftarrow \dfrac{\mathrm{acc}(L_{\mathrm{down}}) - \mathrm{acc}(L_{\mathrm{up}})}{\mathrm{acc}(L_{\mathrm{down}}) + 10^{-12}}$

  if $\Delta_{\mathrm{acc}} \geq \tau_{\mathrm{acc}}$ **then**

    continue

  else

    if $\rho \cdot K(L_{\mathrm{down}}) < K(L_{\mathrm{up}})$ **then**

      swap $S[i]$ **and** $S[i+1]$

    else

      continue

    end if

  end if

end for

return S

itself, its confidence level (measured by PPL) in the annotated task, and a dynamic normalized ranking inspired by Farr [31]. The specific consensus ensemble process is applied to each task j based on the number of available candidate results, following the process below:

1. Uncontested Case: If task j was annotated by only one agent, its result would be accepted as the final result by default.

2. Contested Case: If task j was annotated by two or more agents, the mechanism would employ a two-stage decision:

1) Majority Voting: A majority voting is conducted among all candidate results. If any results is supported by a clear majority of the agents ($> 50\%$), it is selected as the consensus. Since each output includes agent-specific strings such as 'model' and 'confidence' to ensure the highest quality, the final result is taken from the majority group with the lowest PPL, indicating the highest confidence.

2) Weighted Voting: If there is no majority, the mechanism proceeds to a dynamic weighted ensemble, the score for each result is calculated using the formula shown in (3) to (6).

$$\mathrm{Score}_{i,j} = \left(A_{i,j}\right)^{\alpha} \cdot \left(R_{i,j}\right)^{\beta} \cdot \left(C_{i,j}\right)^{\gamma} \tag{3}$$

$$A_{i,j} = \frac{\mathrm{Acc}(L_i)}{\sum_{k \in P_j} \mathrm{Acc}(L_k)} \tag{4}$$

$$R_{i,j} = \frac{n_i - \mathrm{rank}_i\left(\mathrm{PPL}_{i,j}\right) + 1}{n_i}, \; R_{i,j} \in (0, 1] \tag{5}$$

$$C_{i,j} = \frac{\max_{k \in P_j}(\mathrm{PPL}_{k,j}) - \mathrm{PPL}_{i,j}}{\max_{k \in P_j}(\mathrm{PPL}_{k,j}) - \min_{k \in P_j}(\mathrm{PPL}_{k,j})} \tag{6}$$

where, $A_{i,j}$, $R_{i,j}$ and $C_{i,j}$ are the normalized historic accuracy, relative ranking and confidence of agent L_i on task j respectively. $\alpha, \beta, \gamma \geq 0$ are hyperparameters with $\alpha+\beta+\gamma = 1$. P_j is the set of participating agents for task j, and n_i is the total number of tasks labeled by L_i. $rank_i(\cdot)$ ranks all the results of L_i in ascending order by a indicator. $\mathrm{PPL}_{i,j}$ denotes the the output PPL score of task j labeled by L_i.

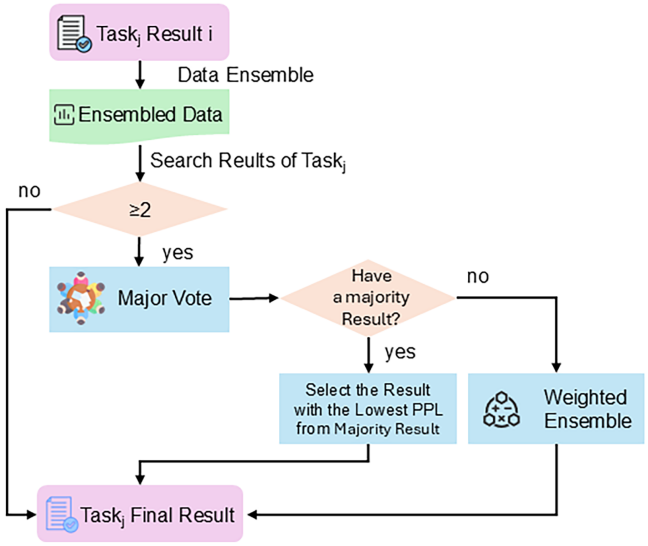

Fig. 2. The process of consensus ensemble mechanism

All indices are normalized so that higher values indicate better quality. For each task j, we sum the normalized scores of each candidate and select the one with the highest total as the final output. The overall framework of the consensus ensemble mechanism is illustrated in Fig. 2. This consensus process provides our complete and final annotated dataset.

IV. Experiments

A. Experimental Setup

1) Scenarios and Datasets: To validate the effectiveness and generalizability of our proposed framework, we designed two distinct experimental scenarios with specific dataset.

a) Scenario 1: Domain-Specific Sequence Labeling for Construction Standards: Scenario 1 focuses on our primary research target: low-cost automated sequence labeling in construction standards for future ACC task, and is was compared with SOTA LLMs. In this study, we created a dataset derived from the NCC and selected 'Section B Structure' volume as a representative example, the raw text was segmented into individual clauses, which serve as the fundamental tasks for annotation. Using the methodology described in Section III-A, the resulting dataset consists of 292 clauses for sequence labeling task to extract and label key entities.

b) Scenario 2: General-Domain Tasks: Scenario 2 evaluates LMABC's generalizability on a standard classification benchmark, the Ideology Books Corpus (IBC) [35], and compares it with a similar SOTA architecture [31]. To isolate the performance differences from framework architecture itself, we run both methods under identical conditions and LLMs.

2) LLM Agents Setup: To evaluate LMABC in the under a lowest-cost setting, we avoid on-prem deployment on large-memory GPUs and instead use free LLM APIs from multiple providers in same machine and network. Across all scenarios, we employ four distinct LLMs as agents: llama-4-Scout-17B-16E-Instruct (L_1), llama-3.3-70b-instruct (L_2), llama-4-maverick-17b-128e-instruct (L_3), and gemini-2.0-flash (L_4) from Together AI, NVIDIA, Cerebras, and Google, respectively. All four models support the 'logprobs' parameter.

Following the methodology detailed in Section III-C, based on preliminary tests, the average input tokens is about 5,500 tokens per task, and the average output tokens is about 12,000 tokens. So we determined the optimal ordering for the agents in the Phase 2 is $[L_1 \Rightarrow L_2 \Rightarrow L_3 \Rightarrow L_4]$, calculated the initial task allocation proportions for Phase 1 (0.15:0.2:0.3:0.35). And then determined hyperparameters in consensus ensemble ($\alpha = 0.7$, $\beta = 0.2$, $\gamma = 0.1$) after multiple tests.

3) Evaluation Metrics: For Scenario 1, We evaluate the performance at two levels to provide an overall assessment.

(1) **Task-level Accuracy:** To evaluate whether the predicted result matches the gold labels, we define y_i as the gold labels and $\hat{y}_i$ as the prediction sequence, as shown in (7).

$$\mathrm{ACC}_{\text{task}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\hat{y}_i = y_i\} \tag{7}$$

(2) **Token-level Metrics:** We compute the micro-averaged Precision, Recall, and F1-score, which are standard for NER tasks to evaluate the performance of the framework, across all tokens from all tasks in the test dataset.

For Scenario 2, following the standard evaluation protocol for tradition classification task, we use the overall accuracy (same as the Task-level Accuracy in (7)) and F1 as the primary metric to compare with other SOTA models.

B. Evaluation Results

1) Scenario 1 Results: To evaluate LMABC, we compared it with multiple individual LLMs. The baselines included the LLMs used in our framework, a suite of mainstream affordable models from OpenAI, and the strongest single model GPT-5. For statistical robustness, each experiences were tested 50 times. Scenario 1 experiment results are shown in Table I.

LMABC achieves a task-level accuracy of 52.62%, a token-level precision 80.44% and F1 score of 78.7%. It outperforms all single LLMs in the LMABC chain (e.g., gemini-2.0-flash: 47.17%), mainstream affordable OpenAI model (e.g., GPT-5-mini: 48.89%) and OpenAI even mainstream o4-mini (52.2%). These results validate the effectiveness of our multi-agent workflow and show that our free solution issuitable than mid-tier paid models for this specialized task. More importantly, LMABC approaches the performance of SOTA model GPT-5 (task-level accuracy of 59.39% and token-level F1 of 84.5%).

Cost efficiency is another major advantage of LMABC. In our setup, it can use free open-source LLMs as agents, making task processing essentially free. Although GPT-5 performs slightly better than our framework, it incurs a much higher cost. Based on the pricing in Table I and per-task token usage mentioned in Section IV-A1, the cost per task can be estimated as USD 0.127. For our dataset of about 300 tasks, the total saving is approximately ΔC = USD $0.127 \times 300 \approx$ USD 38. Therefore, LMABC maintains accuracy close to the strongest baseline, while offering a practical and cost-efficient solution for large-scale automated sequence labeling in building codes, ideal for annotation projects with limited budgets.

2) Scenario 2 Results: The results of LMABC and the compared baselines in Scenario 2 experiment are shown in Table II. As reported in Table 2, LMABC achieves an accuracy of 92.26%, significantly outperforming LLM Chain Ensembles at 86.6%. It also shows lower variance (2.7% vs. 3.9%), indicating more stable performance across repeated trials.

Another notable advantage is cost efficiency. For a fair comparison, we assume the total number of tasks be M, LMABC and the SOTA baseline use the same number of LLM agents, denoted by n. According to Section III-C, the total processed tasks of LMABC is given as follows:

$$Req_{\Phi} = M \ + \ M \cdot \sum_{i=1}^{n-1} Ratio_i \cdot \left(p_i \ + \ \sum_{k=1}^{i-1} \left(\prod_{j=k}^{i-1} Ratio_j \right) p_k \right) \tag{8}$$

From (8), under the constraint on $Ratio_i$, the Req_{Φ} is typically less than 2M. At the same time, the calculation for total processed tasks of SOTA is $\frac{n+1}{2} M$. Therefore, the LLM Chain Ensembles requires approximately 2.5M requests in the experiment. It demonstrates that fewer redundant annotations are sufficient to achieve higher accuracy. This directly translates to a reduction in the total number of tokens consumed and hence lower computational costs.

In summary, Scenario 2 demonstrates that our framework not only outperforms existing ensemble methods in classification accuracy but also reduces the need for redundant annotations, making it a more effective and cost-efficient solution for general sequence labeling and classification tasks.

C. Ablation Experiments

To validate core components of LMABC, we conducted a series of ablation studies on three parts: task allocation, consensus ensemble, and the number of agents.

1) Task Allocation and Consensus Ensemble: To quantify the contributions of task allocation and consensus ensemble, we compare the complete LMABC with three variants: Φ_{Avg} removes the initial task constraint mentioned in Section III-C and assigns tasks evenly and randomly; Φ_{Vote} keeps majority voting only and, when no majority exists, resolves by normalized rank; Φ_{Wei} removes the majority vote and applies the weighted ensemble directly for all tasks.

All experiment results are presented in Table III. The complete LMABC framework achieves the best performance across all metrics, which demonstrates that each component of our design in LMABC contributes to the final performance.

Φ_{Avg} drops most sharply, with task-level accuracy plummeting to 45.82%, highlighting our task allocation mechanism is crucial in the framework. By assigning more capable agents downstream and allocating tasks intelligently, LMABC better handles difficult tasks, thereby boosting the overall accuracy.

TABLE I
SCENARIO 1 RESULTS BY MODEL. PRICES ARE OFFICIAL RATES AS OF NOV. 2025; 'FREE AVAILABLE' DENOTES ZERO-COST AVAILABILITY; SUBSCRIPTS INDICATE STANDARD DEVIATIONS.

	Task Level	**Token Level**			**Price (USD)**
	ACC (%)	Precision (%)	Recall (%)	$F1$ (%)	per 1 million tokens
o4-mini	52.2 ± 1.96	78.53 ± 1.14	75.84 ± 4.57	77.09 ± 2.51	Input: \$1.1 Output: \$4.4
GPT-5	59.39 ± 1.92	83.47 ± 0.55	85.57 ± 1.61	84.5 ± 0.5	Input: \$1.25 Output: \$10
GPT-5-mini	48.89 ± 1.20	69 ± 1.16	79.35 ± 1	73.81 ± 1.04	Input: \$0.25 Output: \$2
GPT-5-nano	30.27 ± 2.34	73.39 ± 2.27	39.87 ± 3.37	51.53 ± 2.43	Input: \$0.05 Output: \$0.4
GPT-4o-mini	40.18 ± 1.80	58.13 ± 3.98	67.12 ± 3.14	62.24 ± 3.23	Input: \$0.15 Output: \$0.6
llama-4-Scout-17B-16E-Instruct	33.31 ± 2.63	56.75 ± 2.70	55.83 ± 1.86	56.27 ± 2.11	Free Available
llama-3.3-70b-instruct	38.01 ± 2.05	64.28 ± 2.96	67.98 ± 1.97	66.05 ± 2.16	Free Available
llama-4-maverick-17b-128e-instruct	47.86 ± 2.86	76.65 ± 1.19	75.57 ± 2.28	76.07 ± 0.86	Free Available
gemini-2.0-flash	47.17 ± 3.83	79.39 ± 1.49	75.23 ± 3.80	77.23 ± 2.62	Free Available
LMABC	**52.62** $\pm$**1.90**	**80.44** $\pm$**1.49**	**77.06** $\pm$**0.45**	**78.70** $\pm$**0.56**	**Free Available**

TABLE II
PERFORMANCE RESULTS FOR EACH MODELS IN SCENARIO 2.

Method	ACC (%)	$F1$ (%)	All Task Requests
Chain Ensembles	86.6 ± 3.9	80.8 ± 1.1	$2.5M$
LMABC	**92.26** $\pm$**2.7**	**90.47** $\pm$**4.3**	$< 2M$

TABLE III
ABLATION RESULTS ON TASK ALLOCATION AND CONSENSUS ENSEMBLE. SUBSCRIPTS INDICATE STANDARD DEVIATIONS.

	Task Level	**Token Level**		
	ACC (%)	Precision (%)	Recall (%)	$F1$ (%)
Φ_{Vote}	50.19 ± 0.77	79.35 ± 0.11	75.47 ± 0.83	77.36 ± 0.49
Φ_{Wei}	50 ± 0.57	79.32 ± 0.08	75.44 ± 0.80	77.33 ± 0.46
Φ_{Avg}	45.82 ± 2.9	74.68 ± 2.11	74.31 ± 1.73	74.48 ± 1.59
Φ	**52.62** $\pm$**1.90**	**80.44** $\pm$**1.49**	**77.06** $\pm$**0.45**	**78.70** $\pm$**0.56**

Φ_{Vote} (ACC 50.19%) and Φ_{Wei} (ACC 50%) are also both under-performing the complete LMABC, indicating that our designed consensus ensemble mechanism is superior to any simplified version. The majority voting efficiently resolves the bulk of uncontested cases, while the weighted voting arbitrates difficult conflicts. The combination of both is the key in achieving the optimal performance.

2) Number of Agents: This section examines the effect of agent number by comparing LMABC with more or fewer LLMs than the four-agent setting used in Section IV-B. We denote $\Phi_{(i)}$ to express that there is i LLMs to form the multi-agent for the LMABC. The experiment results were shown in Fig. 3. Compared with $\Phi_{(4)}$, $\Phi_{(3)}$ removes the low performance LLM 'llama-4-Scout-17B-16E-Instruct', and $\Phi_{(5)}$ adds an affordable LLM 'GPT-4o-mini' produced by OpenAI, which can also support the 'logprob' parameter.

As illustrated in Fig. 3, the results show a consistent increase in Task-Level Accuracy as the number of agents grows from 3 to 5. This validates our fundamental hypothesis that involving more agents in the consensus process enhances the reliability of the final annotations through multi-perspective cross-validation. However, we also can find that the rate of performance growth decelerates. More importantly, increasing

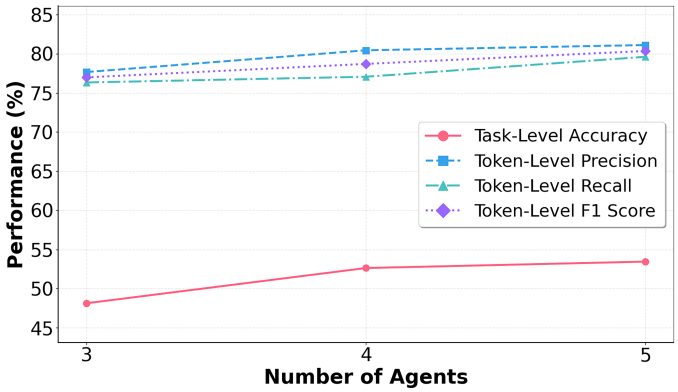

Fig. 3. Ablation study on different number of agents, showing that performance improves across all metrics as the agent count increases from 3 to 5.

the number of agents introduces significant practical costs. A longer agent chain implies greater synchronization delays during Phase 2, which increases the total task processing time due to potential network congestion and API latencies. In our experiments, we also observed a higher risk of system stalls or timeout due to the extended multi-agent chain.

In summary, while adding more agents can theoretically improve accuracy, it leads to diminishing returns in practice while reducing overall system efficiency and stability. Therefore, how to find the best balance between performance benefits and costs (including time costs and computing resources) will be a focus worthy of in-depth research in the future.

V. CONCLUSION

Automated sequence labeling for domain-specific regulatory texts like AEC industry is a critical task for downstream applications like ACC and KG creation, yet its advancement is hindered by the "high-cost, low-resource" dilemma. Traditional supervised methods require lots of expensive annotated data, while strategies relying on a single, SOTA LLM are prohibitively costly. In response, this paper proposed LMABC, a low-cost, low-resource LLM multi-agent framework that orders a team of affordable LLM agents by introducing a two-phase chain structure with task allocation and consensus to balance accuracy and efficiency. Experimental results on both

domain-specific datasets and general benchmarks demonstrate that LMABC achieves competitive accuracy compared to SOTA single LLM while significantly reducing overall cost.

Future work will explore adaptive parameter tuning, dynamical consensus method to reduce redundancy, and retrieval-augmented generation (RAG) combined with ontologies to improve labeling reliability to support domain KG creation.

VI. Acknowledgment

This work is supported by the scholarship from the China Scholarship Council (CSC) while the first author pursues his PhD degree in the University of Wollongong.

References

[1] L. Jiang, J. Shi, and C. Wang, "Multi-ontology fusion and rule development to facilitate automated code compliance checking using BIM and rule-based reasoning," *Advanced Engineering Informatics*, vol. 51, p. 101449, 2022.

[2] D. M. Salama and N. M. El-Gohary, "Semantic Text Classification for Supporting Automated Compliance Checking in Construction," *Journal of Computing in Civil Engineering*, vol. 30, no. 1, p. 04014106, 2016.

[3] H. Wan, J. Zhang, Y. Chen, W. Xu, and F. Feng, "Exploring GenAI applications in building research and industry: A review," *Building Simulation*, vol. 18, no. 6, pp. 1251–1273, 2025.

[4] Y. Zhang and G. Xiao, "Named Entity Recognition Datasets: A Classification Framework," *International Journal of Computational Intelligence Systems*, vol. 17, no. 1, p. 71, 2024.

[5] T. Beach, J. Yeung, N. Nisbet, and Y. Rezgui, "Digital approaches to construction compliance checking: Validating the suitability of an ecosystem approach to compliance checking," *Advanced Engineering Informatics*, vol. 59, p. 102288, 2024.

[6] Z. Zheng, S. Marié, E. Farazdaghi, E. Yahia, K. Makhoul, T. Lagarde, R. E. Meouche, and F. Ababsa, "Mastering building management systems data points tagging with minimal examples: Unveiling the power of large language models," *Energy and Buildings*, vol. 328, p. 115173, 2025.

[7] T. Wang, Y. Zhan, J. Lian, Z. Hu, N. J. Yuan, Q. Zhang, X. Xie, and H. Xiong, "LLM-powered Multi-agent Framework for Goal-oriented Learning in Intelligent Tutoring System," in *Companion Proceedings of the ACM on Web Conference 2025 (WWW '25)*. ACM, 2025, pp. 510–519.

[8] M. Omar, B. S. Glicksberg, G. N. Nadkarni, and E. Klang, "Refining LLMs outputs with iterative consensus ensemble (ICE)," *Computers in Biology and Medicine*, vol. 196, p. 110731, 2025.

[9] C. Sun, S. Huang, and D. Pompili, "LLM-Based Multi-Agent Decision-Making: Challenges and Future Directions," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5681–5688, 2025.

[10] P. Zhou and N. El-Gohary, "Ontology-Based Multilabel Text Classification of Construction Regulatory Documents," *Journal of Computing in Civil Engineering*, vol. 30, no. 4, p. 04015058, 2016.

[11] Q. Cheng, L. Chen, Z. Hu, J. Tang, Q. Xu, and B. Ning, "A novel prompting method for few-shot NER via LLMs," *Natural Language Processing Journal*, vol. 8, p. 100099, 2024.

[12] C. Eastman, J.-m. Lee, Y.-s. Jeong, and J.-k. Lee, "Automatic rule-based checking of building designs," *Automation in Construction*, vol. 18, no. 8, pp. 1011–1033, 2009.

[13] T. Beach, Y. Rezgui, H. Li, and T. Kasim, "A rule-based semantic approach for automated regulatory compliance in the construction sector," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5219–5231, 2015.

[14] M. Aydın, "A Review of BIM-Based Automated Code Compliance Checking: A Meta-Analysis Research," in *Automation and Control - Theories and Applications*. IntechOpen, 2022.

[15] P. Zhou and N. El-Gohary, "Domain-Specific Hierarchical Text Classification for Supporting Automated Environmental Compliance Checking," *Journal of Computing in Civil Engineering*, vol. 30, no. 4, p. 04015057, 2016.

[16] M. Yang, Q. Zhao, L. Zhu, H. Meng, K. Chen, Z. Li, and X. Hei, "Semi-automatic representation of design code based on knowledge graph for automated compliance checking," *Computers in Industry*, vol. 150, p. 103945, 2023.

[17] Z. Ma, H. Zhu, X. Xiang, Ž. Turk, and R. Klinc, "Automatic compliance checking of BIM models against quality standards based on ontology technology," *Automation in Construction*, vol. 166, p. 105656, 2024.

[18] N. Chen, X. Lin, H. Jiang, and Y. An, "Automated Building Information Modeling Compliance Check through a Large Language Model Combined with Deep Learning and Ontology," *Buildings*, vol. 14, no. 7, p. 1983, 2024.

[19] S. Madireddy, L. Gao, Z. U. Din, K. Kim, A. Senouci, Z. Han, and Y. Zhang, "Large Language Model-Driven Code Compliance Checking in Building Information Modeling," *Electronics*, vol. 14, no. 11, p. 2146, 2025.

[20] H. Ghanem and C. Cruz, "Fine-tuning or prompting on LLMs: Evaluating knowledge graph construction task," *Frontiers in Big Data*, vol. 8, p. 1505877, 2025.

[21] M. Neves and U. Leser, "A survey on annotation tools for the biomedical literature," *Briefings in Bioinformatics*, vol. 15, no. 2, pp. 327–340, 2014.

[22] R. R. Slavescu, M. N. Oltean, A. P. Torok, and K. C. Slavescu, "Automatic Learning of Medical Text Annotation Rules – a Case Study on Endoscopies," in *International Conference on Advancements of Medicine and Health Care through Technology; 12th - 15th October 2016, Cluj-Napoca, Romania*. Springer, 2017, vol. 59, pp. 248–251.

[23] D. Zhu and K. W. Wong, "An evaluation study on text categorization using automatically generated labeled dataset," *Neurocomputing*, vol. 249, pp. 321–336, 2017.

[24] J. Peng and X. Liu, "Automated code compliance checking research based on BIM and knowledge graph," *Scientific Reports*, vol. 13, no. 1, p. 7065, 2023.

[25] A. Ramanathan, J. Liang, D. Sommerfield, and H. Azari, "Automatic Data Labeling Using Large Language Models," in *Smart Multimedia*. Springer, 2025, vol. 14957, pp. 241–251.

[26] Z. Tan, D. Li, S. Wang, A. Beigi, B. Jiang, A. Bhattacharjee, M. Karami, J. Li, L. Cheng, and H. Liu, "Large Language Models for Data Annotation and Synthesis: A Survey," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP '24)*. Association for Computational Linguistics, 2024, pp. 930–957.

[27] H. Cai, Y. Li, W. Wang, F. Zhu, X. Shen, W. Li, and T.-S. Chua, "Large Language Models Empowered Personalized Web Agents," in *Proceedings of the ACM on Web Conference 2025 (WWW '25)*. ACM, 2025, pp. 198–215.

[28] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large Language Model Based Multi-agents: A Survey of Progress and Challenges," in *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence (IJCAI '24)*. International Joint Conferences on Artificial Intelligence Organization, 2024, pp. 8048–8057.

[29] J. Gao, B. Chen, X. Zhao, W. Liu, X. Li, Y. Wang, W. Wang, H. Guo, and R. Tang, "LLM4Rerank: LLM-based Auto-Reranking Framework for Recommendations," in *Proceedings of the ACM on Web Conference 2025 (WWW '25)*. ACM, 2025, pp. 228–239.

[30] K. Xiong, X. Ding, Y. Cao, T. Liu, and B. Qin, "Examining Inter-Consistency of Large Language Models Collaboration: An In-depth Analysis via Debate," in *Findings of the Association for Computational Linguistics (EMNLP '23)*. Association for Computational Linguistics, 2023, pp. 7572–7590.

[31] D. Farr, N. Manzonelli, I. Cruickshank, K. Starbird, and J. West, "LLM Chain Ensembles for Scalable and Accurate Data Annotation," 2024.

[32] C. Fang, X. Li, Z. Fan, J. Xu, K. Nag, E. Korpeoglu, S. Kumar, and K. Achan, "LLM-Ensemble: Optimal Large Language Model Ensemble Method for E-commerce Product Attribute Value Extraction," 2024.

[33] J. Qiu, L. Qi, J. Wang, and G. Zhang, "A hybrid-based method for Chinese domain lightweight ontology construction," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 9, pp. 1519–1531, 2018.

[34] H. Zhang and R. Zhang, "Few-shot learning with large foundation models for automated segmentation and accessibility analysis in architectural floor plans," *Journal of Infrastructure Intelligence and Resilience*, vol. 4, no. 2, p. 100137, 2025.

[35] Y. Sim, B. D. L. Acree, J. H. Gross, and N. A. Smith, "Measuring Ideological Proportions in Political Speeches," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*. Association for Computational Linguistics, 2013, pp. 91–101.