ArchiDocGen: Multi-Agent Framework for Expository Document Generation in the Architectural Industry

Junjie Jiang[1,*], **Haodong Wu**[1,*], **Yongqi Zhang**[1,†], **Songyue Guo**[1], **Bingcen Liu**[1], **Caleb Chen Cao**[2], **Ruizhe Shao**[3], **Chao Guan**[3], **Peng Xu**[3], **Lei Chen**[1,2]

[1]The Hong Kong University of Science and Technology (GZ), Guangzhou, China
[2]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[3]China State Construction Engineering (Hong Kong) Limited, Hong Kong SAR, China

Abstract

The architectural industry produces extensive documents, including method statements—expository documents that integrate multi-source data into actionable guidance. Manual drafting however is labor-intensive and time-consuming. This paper introduces ArchiDocGen, a multi-agent framework automating method statement generation. Unlike traditional approaches relying on static templates or single-pass generation, ArchiDocGen decomposes the task into three steps: outline generation, section-based content generation, and polishing, each handled by specialized agents. To provide domain expertise, ArchiDocGen employs a section-based retriever to fetch and synthesize relevant documents from its custom knowledge base. Each section is generated through iterative reasoning of a section-based chain-of-thought (SeCoT) scheme, followed by refinement to meet professional standards. To evaluate the generated method statements, we partner with the industry to establish a multi-dimensional evaluation system by combining automatic and empirical methods. Experiments show that ArchiDocGen achieves 4.38 ContentScore, excelling in specialization, completeness, organization, and clarity. Additionally, a web-based application for ArchiDocGen is developed and deployed with industry partners[1]

1 Introduction

Enterprises in the architectural industry continuously produce extensive documents. Among these, method statements feature well-organized structure and composition logic, integrating multi-source data like project descriptions, work methods, and involved equipments into actionable instructions for site supervisors and workers to execute activities (O'Neill et al., 2022; Borys, 2012). However, drafting such structured method statements is costly. As depicted in the upper part of Figure 1, engineers often spend weeks collecting documents to analyze specific requirements, write step-by-step, and repeatedly proofread for adherence to industry standards. Traditional approaches often involve using static templates filled in manually by engineers (Mi et al., 2018). It lacks the flexibility for varied project demands, limiting efficiency in many cases.

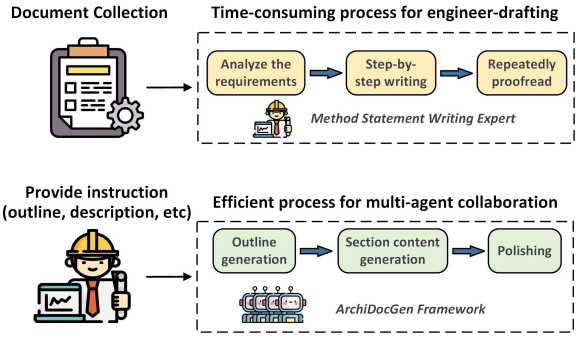


Figure 1: Comparison of the traditional (top) and proposed (bottom) approaches to method statement drafting. The manual approach is a labor-intensive and time-consuming process, while ArchiDocGen uses multi-agent collaboration for automated, efficient generation.

While large language models (LLMs) have achieved broad generative applications across healthcare, finance, and architecture (Yuan et al., 2024; Pu et al., 2024; Wang et al., 2024b), automatically drafting method statements is ineffective due to the specialized knowledge and composition logic required. Consequently, generating professional and specialized method statements and relying solely on direct prompting of a single LLM is difficult (Shao et al., 2024b).

To tackle these challenges, this paper proposes **ArchiDocGen**, a multi-agent framework for expository document generation in the architectural industry. Considering the inherent structure and composition logic in drafting method statements, we decompose the task into three steps: outline

[*] These authors contributed equally.
[†] Corresponding authors: Yongqi Zhang
[1] http://archidocgen.online.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 605–618
July 28-30, 2025

generation, section-based content generation, and polishing. Each step corresponds to an agent with a specific role. Unlike the "Plan-Execute" paradigm that relies on the LLM's inherent knowledge (Bai et al., 2025; Zhang et al., 2024; Li and Zhang, 2024), ArchiDocGen can reference expert-authored method statements from similar projects. Specifically, by extracting metadata such as titles and section content from previous method statements, it constructs a knowledge base aiding the method statement generation. In outline generation, $OutlineAgent$ references method statements on relevant titles to produce a detailed, in-demand outline. In section-based content generation, $SectionAgent$ drafts section content tailored to the project's requirements. It is guided by a section-based chain-of-thought (SeCoT) scheme that prompts $SectionAgent$ to progressively reason what each section should compose. Ultimately, $PolishAgent$ concatenates all sections and polishes the overall method statement to ensure coherence. The whole process mirrors the engineer user's drafting logic. Notably, ArchiDocGen can be generalized to other industrial scenarios with clear structure and composition logic, such as clinical report (Wang et al., 2023), code document (Dvivedi et al., 2024), and financial documentation (Chen et al., 2024). To assess the generated method statements, we partner with industry experts to establish a multi-dimensional evaluation system combining automatic and empirical methods.

Our contributions are summarized as follows:

• We propose a multi-agent generation framework ArchiDocGen that automates method statement generation, enhancing controllability and quality through incorporating domain-specific document composition logic.

• We propose a SeCoT scheme that guides $SectionAgent$ in generating user-specified content by prompting relevant questions and retrieving references, thereby improving specialization.

• To evaluate the quality of the generated method statements, we establish a multi-dimensional evaluation system, providing an example for the evaluation of automatic expository document generation.

2 Related Works

2.1 Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has become a crucial technique for improving factual accuracy in domain-specific document generation (Ji et al., 2023; Zhao et al., 2024). Previous works (Chen et al., 2024; Kwon et al., 2023; Balepur et al., 2023) have demonstrated RAG's effectiveness, especially in generating factually reliable content. Nevertheless, generating a structured and expository document especially for industry practice (e.g. method statement) presents additional challenges beyond mere factual correctness. Expository document generation requires the coherent integration from multi-source references (Balepur et al., 2023). For instance, Shen et al. (2023) utilized the retrieval technique to integrate various sources for structure planning, highlighting the necessity of planning in expository document generation. Chen et al. (2024) adopted graph-based RAG to enhance the logical consistency and quality in report generation of financial market analysis. Balepur et al. (2023) generated expository texts through iteratively combining content planning, fact retrieval, and rephrasing. However, existing methods still struggle to adapt to real-world scenarios due to limited applicability.

2.2 Multi-Agent for Document Generation

Multi-agent systems have demonstrated remarkable potential in document generation fields (Luo et al., 2024; Musumeci et al., 2024; Ramu et al., 2024). Current works primarily utilize a two-stage "Plan-Execute" paradigm, where the planning stage involves agents developing a global understanding of the document generation task (Li and Zhang, 2024; Zhang et al., 2024; Huot et al., 2025). The execution stage then assigns specialized agents to generate detailed, contextually precise contents (Luo et al., 2024). For instance, Huot et al. (2025) applied multi-agent systems for story generation. In the planning phase, multiple agents collaborate to draft task descriptions and plot elements, incorporating a human-in-the-loop mechanism to guide and adjust the process. This approach resembles the method proposed by Jiang et al. (2024), where human oversight helps fine-tune the discourse generated by LLMs. Bai et al. (2025) also employed a plan-execute approach, exploring and validating the capability of LLMs to generate exceptionally long texts. Despite these successes, current multi-agent methods primarily focus on open-domain document generation and often fail to adapt to industry-specific practices. Our work enhances the plan-execute paradigm, which integrates domain-specific composition logic, ensures controllable document generation and produces specialized,

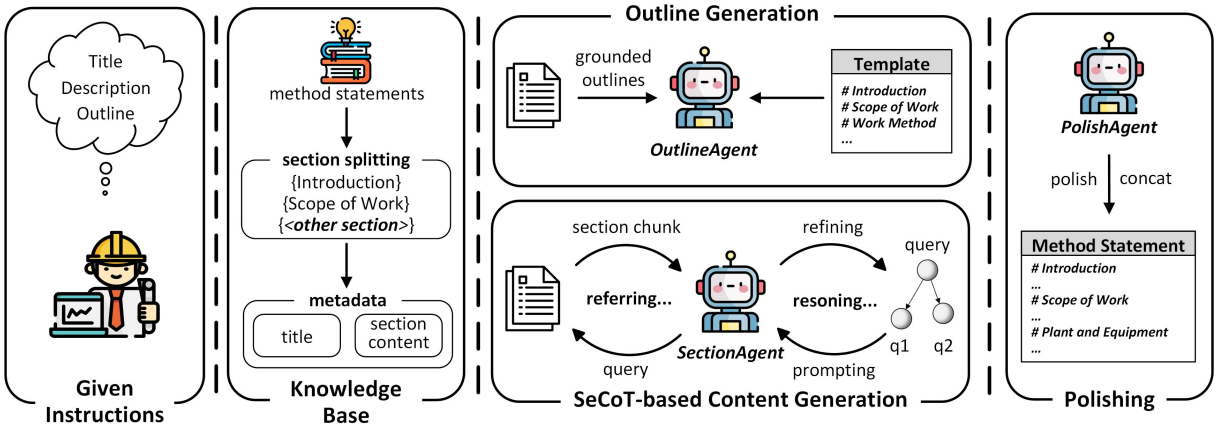


Figure 2: Overview of the ArchiDocGen framework for automated method statement generation. Starting from user-provided instructions, the framework proceeds with building knowledge base, outline generation by the OutlineAgent, section content generation with the SectionAgent using SeCoT, and final refinement by the PolishAgent, resulting in a professional and specified method statement.

convincing expository documents.

3 Methodology

3.1 Framework Overview

Our proposed **ArchiDocGen** framework is shown in Fig. 2. It begins with a provided title T and a brief description D, following a modular pipeline, i.e., outline generation, section content generation, and polishing.

• **Outline Generation**. *OutlineAgent* creates a fine-grained outline (i.e. multi-level section headings) based on reference outlines of similar titles. It decides the logical composition of the targeted method statement.

• **Section Content Generation**. $SectionAgent$ utilizes a section-based chain-of-thought (SeCoT) scheme to progressively reason and draft section content tailored to project requirements.

• **Polishing**. $PolishAgent$ is tasked with concatenating all the generated section contents and refines them, enhancing the readability and overall quality of the final method statement.

Formally, the entire process to generate a method statement $\mathcal{M} = \{s_k \in S \mid k = 1, 2, \ldots, n\}$ with n sections can be formulated as:

$$\mathcal{M} = ArchiDocGen(T, D, O, \mathcal{V}) \quad (1)$$

where *ArchiDocGen*$(\cdot)$ represents the proposed framework, T, D, O, and $\mathcal{V}$ denote the provided title, description, outline template, and knowledge base, respectively. The entire process is shown in Algorithm 1. The process starts with the *OutlineAgent*, which creates a structured outline from the provided inputs. For each section heading h covering in the generated outline O_{gen}, the *SeCoT* scheme is invoked to generate the section content s. Once all section contents are generated, M_{draft} with these sections is then concatenated and polished by *PolishAgent*.

Algorithm 1 Generation Process of ArchiDocGen Framework

Input: Title T, Description D, Reference Outline O, Requirements $\mathcal{R}$, Knowledge Base $\mathcal{V}$
Output: Method Statement M_{draft} (a list of section contents)

$list : M_{draft} \leftarrow \varnothing$
$O_{gen} \leftarrow OutlineAgent(T, D, O, \mathcal{V})$
for each section heading $h_k \in O_{gen}$ **do**
  $s_k \leftarrow SeCoT(T, D, h_k, \mathcal{R}, \mathcal{V})$
  $M_{draft} \leftarrow M_{draft} \cup s_k$
end for
$M_{draft} \leftarrow PolishAgent(M_{draft})$
return M_{draft}

3.2 Outline Generation

An appropriate outline reflects a document's composition logic. Directly prompting an LLM to generate outline without clear references may result in deviations, negatively affecting subsequent section content. Therefore, we provide the *OutlineAgent* with an outline template O_{temp} containing generic-level sections. However, solely relying on

this static template restricts the method statement's adaptability. To overcome this limitation, we split the expert-authored method statements into sections to form a knowledge base. Then, the retriever recalls grounded documents on relevant title, i.e. method statements from similar projects, which denoted as M. The section headings h are extracted from these documents to create reference outlines $\mathcal{O}_{ref}$, which $OutlineAgent$ uses to generate the targeted outline O_{gen}. The process can be formulated as below:

$$\mathcal{O}_{ref} = \{h_j \mid j \in \text{Top}_k\left(\text{sim}\left(Q, M_j\right)\right), M_j \in \mathcal{V}\} \quad (2)$$

$$O_{gen} = OutlineAgent(O_{temp}, \mathcal{O}_{ref}) \quad (3)$$

where Q denotes the user query, i.e. the title-description pair $Q = (T, D)$, $\text{Top}_k(\text{sim}(Q, M_j))$ represents the indices of the top k reference method statements, h_j means the section headings extracted from reference M_j.

3.3 SeCoT-based Generation

In this module, content is generated section by section, with each section referencing relevant sections from retrieved documents. Inspired by the iterative retrieval methods (Press et al., 2023; Shao et al., 2023), we employ a multi-step and section-based chain-of-thought (SeCoT) scheme to iteratively refine queries, enabling the system to progressively focus on detailed and relevant information for content generation. Algorithm 2 shows the generation process. It consists of two primary branches: direct generation (lines 2–4) and section-based chain-of-thought (SeCoT) (lines 7–13). Here we focus on the SeCoT scheme, and the former is detailed in Appendix A.1. As shown in Fig. 3, the iterative process starts by querying a vectorized knowledge base. For instance, if the target method statement is titled "Concrete Curing", method statements related to "Concrete Curing" are retrieved. For a specific section like "Work Method", section chunks with similar headings are extracted from these documents. This ensures only the most contextually relevant information is used for subsequent reasoning. Specifically, the retrieved section chunks serve as references for the *SectionAgent*, which then iteratively refines the queries through the SeCoT process. Each iteration produces an increasingly targeted query, facilitating the retrieval of detailed information and enabling the generation of sound, applicable section content. The iterative loop continues until a predefined maximum iteration count

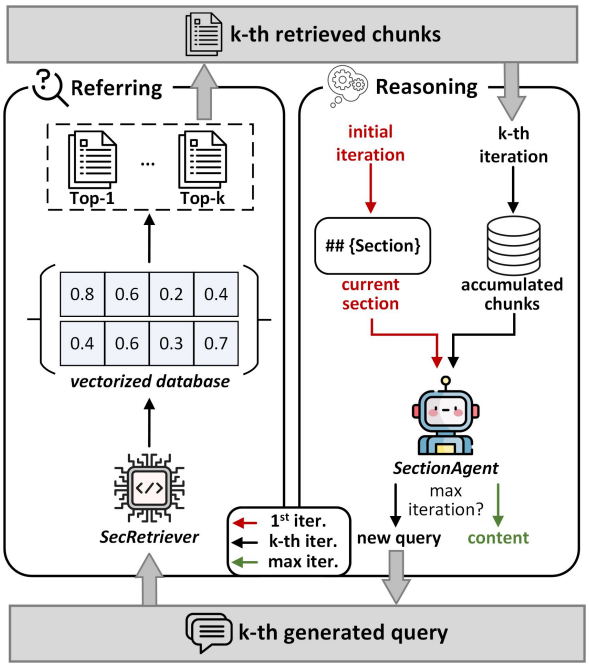


Figure 3: The SeCoT-based generation process includes referring to retrieved section chunks (left) and reasoning through multi-step queries (right), accumulating background knowledge to generate specialized section content.

is reached. Once the accumulated chunks are synthesized, *SectionAgent* generates the final content for the current section. In addition, we introduce plug-and-play rules, referred to as implicit standards, that the document generation task must follow. These rules also inherently reflect the composition logic of the document. For details, see Appendix A.2.

3.4 Polishing

The initial draft of the method statement, created by concatenating all sections, often lacks smooth transitions and coherence. Additionally, the generated content may include structured elements like markdown-tables, which are sometimes incomplete and cause rendering issues. We prompt $PolishAgent$ to processes all concatenated sections to ensure seamless transitions, eliminate duplicates, fix incomplete markdown-tables, and preserve a clear hierarchy in the method statement.

4 Implementation and Evaluation

4.1 Dataset Preparation

To construct a robust and comprehensive knowledge base, we collaborate with architectural industry partners to gather 1200 real-world expert-

Algorithm 2 Generation Process of k^{th} Section Content

Input: Title and Description Q_0, Section Heading $h_k \in O_{gen}$, Requirements $\mathcal{R}$, Template-driven Generation $\mathcal{N}_t$

Output: s_k

$h_k^c \leftarrow Classify(h_k)$
if $h_k^c \in \mathcal{N}_t$ **then**
  return $s_k \leftarrow Direct_{gen}(h_k^c)|Ext(Q_0, h_k^c)$
end if
$list: \bar{\mathcal{C}} \leftarrow \varnothing$
$list: Q \leftarrow Q_0$
while $i < max\_iter$ **do**
  $q \leftarrow SeCoT(Q, Q_0, h_k, \mathcal{R}[h_k^c])$
  $\mathcal{C}_{ref} \leftarrow retrieve(Q)$
  $Q \leftarrow append(Q, q)$
  $\bar{\mathcal{C}} \leftarrow append(\bar{\mathcal{C}}, \mathcal{C}_{ref})$
  $i \leftarrow i + 1$
end while
$s_k \leftarrow SectionAgent(\bar{\mathcal{C}}, Q_0, h_k, \mathcal{R}[h_k^c])$
return s_k

authored method statements. The collected method statements cover various architectural activities, such as concrete pouring, and scaffolding operations. These documents are actual field materials used by certified engineers across multiple architectural projects, making its quality, structure, and domain coverage ensure its representativeness. Furthermore, each document contains detailed procedural knowledge, with an average of over 28 section-level units per article (see Table 1), resulting in a rich and dense knowledge base. The collected documents are scanned PDFs,

	Dataset Statistics	Value
article-wise	Average Amount of All-level Sections	28.5
	Average Word Count of a Section	152.2
	Average Word Count of Whole Document	2895.5
outline-wise	Average Amount of First-level Heading	12.9
	Average Amount of Second-level Heading 2	10.6
	Average Amount of Third-level Heading	4.7

Table 1: Dataset Statistics of human-authored method statements.

typically structured into sections such as Introduction, Scope of Work, Work Method, etc. Subsequently, we adopt the end-to-end document extraction tool MinerU (Wang et al., 2024a) to recognize the collected PDFs. This tool effectively parses the scanned PDFs through layout detection, table recognition, and text extraction. The parsed PDFs are converted into markdown-formatted documents. However, the initial extracted texts contain noise, e.g., redundant empty lines, formatting inconsistencies, etc. We employ gpt-4o-0806 for data cleaning and alignment, thereby restoring the original content integrity. The processed markdown texts are then parsed into hierarchical section-based chunks, which are stored as a structured knowledge base to facilitate efficient retrieval.

4.2 Automatic Metrics

In addition to ROUGE and BERTScore, we also employ the following automatic metrics for generated method statements:

OutlineScore: A five-point scale on *clarity*, *completeness*, *organization*, and *specialization* for the outline quality using gpt-4o-0806. *N-shot* examples from human-written method statements are provided to align with expert judgement during evaluation, the evaluation instruction is shown in Appendix E.2.

ContentScore: It evaluates the generated method statements by assessing individual section content quality with gpt-4o-0806, incorporating expert-defined criteria, redundancy penalties, and a threshold for minimum required sections to ensure fairness and comprehensiveness. More details see Appendix B.

Evaluator LM: To mitigate bias in gpt-4o-0806 scoring, we also use a third-party evaluator prometheus-7b-v2.0 (Kim et al., 2024), which is exclusively fine-tuned to align with human judgment. We adopt it to assess the generated method statements over the expert-defined criteria. The criteria is defined in Appendix E.3.

4.3 Expert Evaluation

Since the real-world evaluation of method statements is primarily empirical-based, we select five experienced industry experts to participate in the evaluation. To facilitate the process, we develop a tailored platform, detailed in Appendix D, to present pairs of generated method statements under different settings. During this, experts score the method statements using the same five-point scale for *clarity*, *completeness*, *organization*, and *specialization*.

	ROUGE-1			ROUGE-2			ROUGE-L			BERTScore	ContentScore	Length	Evaluator LM
	P	R	F1	P	R	F1	P	R	F1				
DeepSeek	**0.70**	0.18	0.27	**0.31**	0.06	0.10	**0.44**	0.09	0.14	**0.78**	2.42	468.3	3.2
GPT-4o-0806	0.69	0.19	0.29	0.21	0.05	0.08	0.31	0.08	0.13	**0.78**	2.58	455.0	3.4
LongWriter	0.47	0.48	0.46	0.15	0.15	0.14	0.16	0.18	0.17	**0.78**	3.91	3378.9	3.9
STORM	0.58	**0.55**	**0.56**	0.24	**0.24**	0.23	0.23	0.25	**0.23**	0.75	2.14	**3913.1**	3.7
ArchiDocGen	0.57	0.53	0.54	0.21	**0.24**	0.21	0.17	**0.29**	0.21	0.76	**4.38**	3240.3	**4.3**

Table 2: Performance Comparison of Different Methods.

5 Experiments

5.1 Main Results

Content Evaluation. To ensure a fair comparison, we use the same retrieval configuration across all baseline methods. From Table 2, it can be observed that there is a significant gap between precision and recall in the direct prompting methods. This difference arises because the directly generated content is too short. Although the semantic vector is close to that of other methods, the textual overlap between the generated content and the references is relatively low (See its ROUGE-F1 values). In contrast, for multi-agent approaches, this gap is reduced, indicating that such methods indeed improve relevance. Moreover, merely ROUGE and BERTScore cannot fully represent the "precision" or "quality" of the generated documents (Bhandari et al., 2020; Zhao et al., 2023). From metrics of both ContenScore and Evaluator LM, our method shows improvements over the baselines, achieving scores of 4.38 and 4.3, respectively.

Furthermore, we also evaluate the outline generation results. As shown Figure 5(a), the directly generated outlines tend to be more clarified (see GPT-4o, DeepSeek). It can be observed in Figure 5(b) that they generally lack second- and third-level headings, which leads to higher scores in *clarity* and *organization*. However, in terms of *specialization* and *completeness*, our method achieves the highest scores. This is also reflected in the distribution of the generated outlines—our method produces outlines that are most similar to human-written ones. This further demonstrates the effectiveness of our approach in outline generation.

5.2 Ablation Study

We conducted an ablation study on ArchiDocGen with its variants: "w/o Outline", "w/o SeCoT", and "w/o Req":

1) "w/o Outline": The variant "w/o Outline" generates method statements without a defined fundamental structure.

2) "w/o SeCoT": The variant "w/o SeCoT" denotes that the whole generation process does not involve multi-step reasoning to produce specified contents.

3) "w/o Req": The variant "w/o Req" denotes that the essential information required by engineers is omitted without the requirements constraints.

From the Table 3, the "w/o Outline" setting, we observe that the generated text length nearly doubles. However, both ROUGE scores and OutlineScore decrease. This indicates that removing this component significantly reduces the content relevance and outline quality. For the two ablations related to content generation (see Table 4), "w/o SeCoT" also leads to a decrease in content relevance, resulting in a substantial drop in the ContentScore. This suggests that deeper reasoning helps improve information recall, thereby making the generated content more specialized and relevant. On the other hand, in the "w/o Req" setting, although the amount of recalled information increases, both ContentScore and the generated text length decrease. This implies that without implicit requirements as guidance, the agent tends to overlook key domain-specific standards. This observation is further supported by the human evaluation results in Figure 4.

	ROUGE			Outline Score	Section Amount	Length
	R-1	R-2	R-L			
ArchiDocGen	0.54	0.21	0.21	4.03	28.5	3240.3
w/o Outline	0.50	0.23	0.20	3.77	48.2	6164.3

Table 3: Comparison of ArchiDocGen with its ablation variant in outline generation.

5.3 Expert Evaluation

Fig. 4 illustrates the expert evaluation results, including the performance between ArchiDocGen

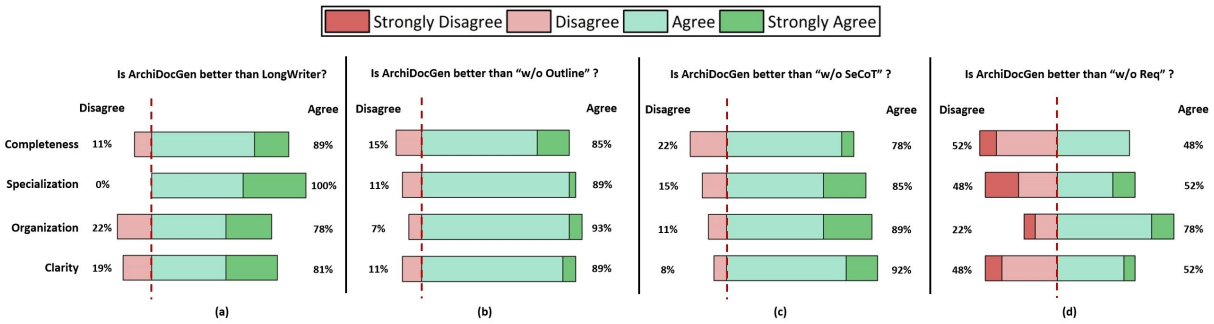


Figure 4: Expert evaluation results comparing ArchiDocGen with the best baseline LongWriter (from the ContentScore metric) and its ablation variants across four dimensions: Completeness, Specialization, Organization, and Clarity.

	ROUGE			Content Score	Section Amount	Length
	R-1	R-2	R-L			
ArchiDocGen	0.54	0.21	0.21	4.38	28.5	3240.3
w/o SeCoT	0.48	0.16	0.16	3.90	32.3	3331.6
w/o Req	0.55	0.27	0.26	4.01	25.3	2675.6

Table 4: Comparison of ArchiDocGen with its ablation variants in section content generation.

and the best-performing baseline LongWriter, and the impact of ArchiDocGen's key components on four dimensions. To ensure the reliability of the blind evaluation, we calculated the Fleiss Kappa (Fleiss, 1971) coefficients for the four comparison groups (i.e., Fig. 4 (a) - (d)), which were 0.64, 0.55, 0.62, and 0.33, respectively. These values indicate substantial, moderate, substantial, and fair agreement levels, demonstrating a generally consistent evaluation among experts. Fig. 4 (a-c) demonstrate that our method significantly outperforms LongWriter, as well as "w/o Outline" and "w/o SeCoT" variants. Additionally, in Fig. 4 (d), 78% of experts agreed that ArchiDocGen performed better in *organization*. However, agreement on the other three dimensions was relatively lower, indicating that requirement constraints played a slightly weaker role in these dimensions but were still essential for maintaining content relevance and logical flow. Experts noted that while the "w/o Req" variant produced shorter content (refer to Table 4), it often omitted critical information. In contrast, ArchiDocGen effectively incorporated requirements to generate more comprehensive and applicable content.

6 Conclusion

In this paper, we introduce ArchiDocGen framework, a multi-agent framework designed to automate and enhance the generation of method statements in the architectural industry. Firstly, our system leverages composition logic to ensure that the generated outline aligns closely with engineer-specified requirements. We incorporate a section-based chain-of-thought scheme to expand and refine queries, thereby enhancing the retrieval of more relevant section chunks. Furthermore, we introduce a detailed section-based evaluation system and incorporate a score penalty mechanism to rectify false generations. To validate our approach, we compare direct prompting and several other multi-agent frameworks on document generation tasks using engineer-specified requirements. We also conducted multi-dimensional manual evaluations of different modules integrated into our system. The results demonstrate that the proposed ArchiDocGen framework effectively generates well-structured, professional method statements.

Limitations

Several limitations of our work are identified through practical industrial feedback.

• High dependence on knowledge base: Since ArchiDocGen depends on previously authored method statements, the absence or limited availability of high-quality reference documents in certain emerging engineering projects may negatively impact the effectiveness.

• Hallucinations and Inaccurate Content: ArchiDocGen powered by LLMs makes it susceptible to common LLM-related issues such as hallucinations and inaccurate content generation. Although the SeCoT approach mitigates these concerns through iterative querying and referencing retrieved section chunks, there is still a risk of generating content that may not fully meet industry common sense

without human validation.

• Difficult to Evaluate: Current evaluation methods of mainstream document generation primarily rely on human evaluation, which introduces subjectivity. Future work can focus on reducing dependence on knowledge bases, improving content accuracy through advanced validation mechanisms, and developing more objective evaluation methods for document generation.

Ethics Statement

Our work adheres strictly to the ethical guidelines throughout the development and deployment. The data utilized in this work are provided by the collaborating architectural enterprise, with explicit approval and clear understanding of the intended research usage. To safeguard privacy and confidentiality, all sensitive information are anonymized before inclusion in our knowledge base. Moreover, we acknowledge the broader implications of generative content tool, such as potential impacts on employment within the industry. We unanimously agree that the developed system is positioned explicitly as an assistive tool, designed to enhance the productivity and efficiency of professionals rather than to replace human. Finally, our work is integrated into proprietary industry systems, and access to the full operational version is currently restricted exclusively to authorized users. To protect the interests of our industry partner, we do not plan to publicly release the developed system.

Acknowledgments

Yongqi Zhang's work is supported by Guangdong Basic and Applied Basic Research Foundation 2025A1515010304, and Guangzhou Science and Technology Planning Project 2025A03J4491. Lei Chen's work is partially supported by National Key R&D Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, Guangdong-Hong Kong Technology Innovation Joint Funding Scheme Project No. 2024A0505040012, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Guangdong Province Science and Technology Plan Project 2023A0505030011, Guangzhou municipality big data intelligence key lab, 2023A03J0012, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab and 2023 HKUST Shenzhen-Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund, from Shui On Xintiandi and the InnoSpace GBA.

References

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. Longwriter: Unleashing 10,000+ word generation from long context llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Nishant Balepur, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Expository text generation: Imitate, retrieve, paraphrase. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 11896–11919. Association for Computational Linguistics.

Manik Bhandari, Pranav Narayan Gour, Atabak Ashfaq, Pengfei Liu, and Graham Neubig. 2020. Re-evaluating evaluation in text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9347–9359. Association for Computational Linguistics.

David Borys. 2012. The role of safe work method statements in the australian construction industry. *Safety Science*, 50(2):210–220.

Yuemin Chen, Feifan Wu, Jingwei Wang, Hao Qian, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, and Meng Wang. 2024. Knowledge-augmented financial market analysis and report generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 1207–1217. Association for Computational Linguistics.

Shubhang Shekhar Dvivedi, Vyshnav Vijay, Sai Leela Rahul Pujari, Shoumik Lodh, and Dhruv Kumar. 2024. A comparative analysis of large language models for code documentation generation. In *Proceedings of the 1st ACM International Conference on AI-Powered Software, AIware 2024, Porto de Galinhas, Brazil, July 15-16, 2024*. ACM.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Fantine Huot, Reinald Kim Amplayo, Jennimaria Palomaki, Alice Shoshana Jakobovits, Elizabeth Clark,

and Mirella Lapata. 2025. Agents' room: Narrative generation through multi-step collaboration. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Yucheng Jiang, Yijia Shao, Dekun Ma, Sina J. Semnani, and Monica S. Lam. 2024. Into the unknown unknowns: Engaged human learning through participation in language model agent conversations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 9917–9955. Association for Computational Linguistics.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 4334–4353. Association for Computational Linguistics.

Deuk Sin Kwon, Sunwoo Lee, Ki Hyun Kim, Seojin Lee, Taeyoon Kim, and Eric Davis. 2023. What, when, and how to ground: Designing user persona-aware conversational agents for engaging dialogue. In *Proceedings of the The 61st Annual Meeting of the Association for Computational Linguistics: Industry Track, ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 707–719. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kunze Li and Yu Zhang. 2024. Planning first, question second: An llm-guided method for controllable question generation. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 4715–4729. Association for Computational Linguistics.

Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024. RepoAgent: An LLM-powered open-source framework for repository-level code documentation generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 436–464, Miami, Florida, USA. Association for Computational Linguistics.

Lin Mi, Chuanrong Li, Peng Du, Jiajia Zhu, Xinfang Yuan, and Ziyang Li. 2018. Construction and application of an automatic document generation model. In *26th International Conference on Geoinformatics, Geoinformatics 2018, Kunming, China, June 28-30, 2018*, pages 1–6. IEEE.

Emanuele Musumeci, Michele Brienza, Vincenzo Suriani, Daniele Nardi, and Domenico Daniele Bloisi. 2024. Llm based multi-agent generation ofnbsp;semi-structured documents fromnbsp;semantic templates innbsp;thenbsp;public administration domain. In *Artificial Intelligence in HCI: 5th International Conference, AI-HCI 2024, Held as Part of the 26th HCI International Conference, HCII 2024, Washington, DC, USA, June 29–July 4, 2024, Proceedings, Part III*, page 98–117, Berlin, Heidelberg. Springer-Verlag.

Cameron O'Neill, Vinod Gopaldasani, and Robyn Coman. 2022. Factors that influence the effective use of safe work method statements for high-risk construction work in australia–a literature review. *Safety science*, 147:105628.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Hongxu Pu, Xincong Yang, Jing Li, and Runhao Guo. 2024. Autorepo: A general framework for multimodal llm-based automated construction reporting. *Expert Syst. Appl.*, 255:124601.

Pritika Ramu, Pranshu Gaur, Rishita Emandi, Himanshu Maheshwari, Danish Javed, and Aparna Garimella. 2024. Zooming in on zero-shot intent-guided and grounded document generation using LLMs. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 676–694, Tokyo, Japan. Association for Computational Linguistics.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. 2024a. Assisting in writing Wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278, Mexico City, Mexico. Association for Computational Linguistics.

Yijia Shao, Yucheng Jiang, Theodore A. Kanell, Peter Xu, Omar Khattab, and Monica S. Lam. 2024b. Assisting in writing wikipedia-like articles from scratch

with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 6252–6278. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Zejiang Shen, Tal August, Pao Siangliulue, Kyle Lo, Jonathan Bragg, Jeff Hammerbacher, Doug Downey, Joseph Chee Chang, and David Sontag. 2023. Beyond summarization: Designing ai support for real-world expository writing tasks. *CoRR*, abs/2304.02623.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, et al. 2024a. Mineru: An open-source solution for precise document content extraction. *CoRR*.

Siyuan Wang, Zheng Liu, and Bo Peng. 2023. A self-training framework for automated medical report generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 16443–16449. Association for Computational Linguistics.

Ziao Wang, Xiaofeng Zhang, and Hongwei Du. 2024b. Mutual information guided financial report generation with domain adaption. *IEEE Trans. Emerg. Top. Comput. Intell.*, 8(1):627–640.

Dong Yuan, Eti Rastogi, Gautam Naik, Sree Prasanna Rajagopal, Sagar Goyal, Fen Zhao, Bharath Chintagunta, and Jeff Ward. 2024. A continued pretrained LLM approach for automatic medical note generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Short Papers, NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 565–571. Association for Computational Linguistics.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. 2024. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10836–10863, Miami, Florida, USA. Association for Computational Linguistics.

Wei Zhao, Michael Strube, and Steffen Eger. 2023. DiscoScore: Evaluating text generation with BERT and discourse coherence. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3865–3883, Dubrovnik, Croatia. Association for Computational Linguistics.

Xinyang Zhao, Xuanhe Zhou, and Guoliang Li. 2024. Chat2data: An interactive data analysis system with rag, vector databases and llms. *Proc. VLDB Endow.*, 17(12):4481–4484.

A Additional Generation Details

A.1 Direct Generation

During the document content generation process, not all sections require the SeCoT process. For certain sections (e.g., "Introduction") , it's sufficient to directly generate content based on the targeted document's title and description according to a predefined template (refer to Appendix E.1). This practice aligns with common document preparation scenarios in various industries.

A.2 Implicit Standards in Section Generation

For implicit standard, we apply $\mathcal{R}$, a set of requirements specifying essential sections across various method statements. These requirements are further categorized into distinct groups, resulting in $\bar{\mathcal{R}} = \{\bar{r}_i \mid i = 1, 2, \ldots, m\}$, where each $\bar{r}_i$ represents a specific category. Then we prompt a LLM to map each section heading h_k in O_{gen} to the most relevant requirement group within $\bar{\mathcal{R}}$, formulated as $\mathcal{R}[h_k^c]$. This ensures precise requirement fragments are accessible during SeCoT-based content generation, aiding the LLM in producing targeted outputs.

B Grading System of ContentScore

The grading process begins by segmenting and categorizing the generated sections, similar to Section 3.3. Each section is evaluated by the LLM based on predefined criteria and *n-shot* examples from human-authored content, producing a list of pairs (section_class, score). Scores within the same category are aggregated, represented as $\mathcal{M}^a$, where each category $s \in \mathcal{M}^a$ corresponds to its aggregated values r. However, it may cause a limitation: some documents score highly for individual sections but misses key sections, lacking fairness and structural completeness.

To address this, industry experts highlight two critical considerations: **1) Critical sections matter the most**, especially sections like "Work Methods," where redundancy is unacceptable. **2) Content and completeness are equally important**, as missing sections significantly reduce quality. Based on these insights, we refine the scoring mechanism as shown in Algorithm 3:

• **Redundancy detection for critical sections**: If a section is repeated, the average score is calculated with a penalty term $1/\sqrt{\text{times}}$, where "times" denotes the repetition count (see line 9). For critical sections, a stricter penalty of $1/\text{times}'$ is applied (see line 11).

• **Completeness of the method statement**: We set a threshold l, which defines as half the average number of sections in human-authored method statements. Generated documents fail to meet this threshold are deemed structurally incomplete (see line 13).

Algorithm 3 The calculation of *ContentScore*

Input: A score set for sections is denoted as $\mathcal{M}^a$, predefined length l, section's category s, scores with the same category r, critical section categories $\mathcal{K}_t$, section repeat times $times$, critical section repeat times $times'$.

Output: $score$

1: $avg \leftarrow 0$
2: $times \leftarrow 0$
3: $times' \leftarrow 1$
4: $set : v \leftarrow \varnothing$
5: **for** (s, r) in $\mathcal{M}^a$ **do**
6: $\quad v \leftarrow add(v, s)$
7: $\quad times \leftarrow len(r)$
8: $\quad$ **if** $s \in \mathcal{K}_t$ **then**
9: $\quad\quad times' \leftarrow times' + times - 1$
10: $\quad$ **end if**
11: $\quad avg \leftarrow avg + \frac{Average(r)}{\sqrt{times}}$
12: **end for**
13: $score \leftarrow \frac{avg}{times'}$
14: **if** $len(v) < l$ **then**
15: $\quad score \leftarrow \frac{score}{l}$
16: **end if**
17: **return** $score$

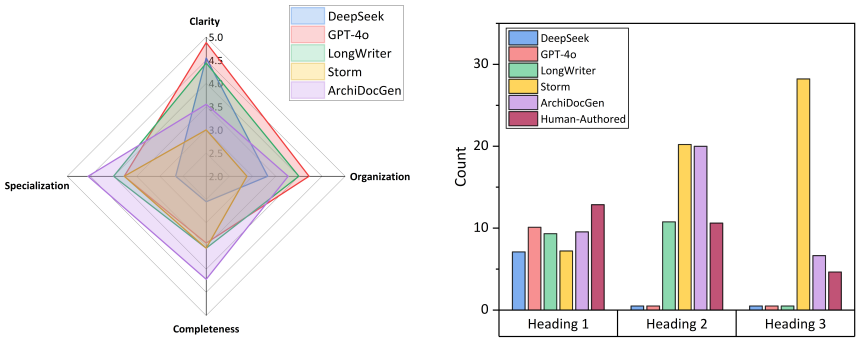
C Experimental Setup

C.1 Baselines

We summarize the comparison of different document generation methods in Table 5. Conditional generation (CG) indicates the document is generated under conditional constraint, while open document generation (ODG) means open-ended. We select several representative document generation methods from Table 5 (e.g., LongWriter, Storm) for experimentation.

C.2 Main Experiment Setup

We use 2*A100 GPUs with 80GB memory for deployment. We select 61 engineering titles from the dataset for the subsequent experiments. The foundation model of ArchiDocGen and Storm powers by DeepSeek-V2.5, while LongWriter



(a) (b)

Figure 5: Dimension evaluation of generated outlines across four dimensions, comparing ArchiDocGen with baselines and human-authored outlines; (b) Section heading statistics for different heading levels, comparing ArchiDocGen and baselines.

	Method	RAG	CoT	CG/ODG
Others	FinanceReport (Chen et al., 2024)	-	-	CG
Direct Prompting	LongWriter (Bai et al., 2025)	-	-	ODG
	MM-PAW (Ramu et al., 2024)	✓	-	ODG
Multi-Agent	PAD-Gen (Musumeci et al., 2024)	-	-	CG
	Agents Room (Huot et al., 2025)	-	-	ODG
	Storm (Shao et al., 2024a)	✓		ODG
	Co-Storm (Jiang et al., 2024)	✓	-	ODG
	ArchiDocGen	✓	✓	CG

Table 5: Comparison of different document generation methods.

means longwriter-glm4-9b. ArchiDocGen employs FAISS[2] for indexing section contents. For vector injection, we use a combination of BCE's[3] embedding-base and reranker-base modules. In outline generation, a Top-k strategy is adopted with $k = 4$. During the SeCoT process, the framework permits a maximum of 3 iterations for reasoning, with each reasoning cycle retrieving the top $k = 2$ chunks. For the OutlineScore and ContentScore evaluations, we utilize 3-shot examples as references. During Evaluator LM judgment, i.e. prometheus (Kim et al., 2024), the expert-defined criteria are adopted (see Appendix E.3).

D Demo and Evaluation Platform

• **Demo.** Our demo allows engineers to input a title, brief description, and optional outline for the desired method statement. ArchiDocGen uses these inputs to first create the document's structure, then generate section contents. As shown in Fig. 6, the system supports post-generation refinement, offering four functions: **1) Modify** for editing specific sections, **2) Delete** for removing irrelevant or redundant sections, **3) Polish** for enhancing overall quality, and **4) Feedback** for suggesting improvements to continuously enhance the system. Additionally, ArchiDocGen supports exporting generated Markdown-based method statements into custom Word templates using Pandoc [4], ensuring compliance with corporate or project. The demonstration of this interactive generation process can be seen through https://youtu.be/PvsjOCzau9U [5].

• **Evaluation Platform.** To ensure unbiased and practical feedback, we developed an evaluation platform (see Fig. 7) for engineers to assess generated method statements. The platform presents side-by-side comparisons of method statements (e.g., ArchiDocGen vs. LongWriter), randomly distributed to avoid bias. Engineers evaluate statements across four dimensions—*clarity*, *organization*, *specialization*, and *completeness*—using a four-point scale ("Strongly Disagree" to "Strongly Agree"). Then, the engineer are asked whether the left side is better than the right side on these dimensions, and they make their own choices based on this premise. To ensure fairness, the source of the statements is hidden, and engineers are only told the content is AI-generated. This blind evaluation approach prevents preconceived notions from influencing their assessments. Pairings and presentations are randomized, and the platform iteratively presents new pairs for unbiased review.

[2] https://github.com/facebookresearch/faiss

[3] https://github.com/netease-youdao/BCEmbedding

[4] https://pandoc.org/

[5] You may notice that the provided demo differs from the version in the video. This is because, as mentioned in our ethics statement, the data and deployment environment are proprietary. To protect the interests of all parties, we have chosen to demonstrate a simplified version.

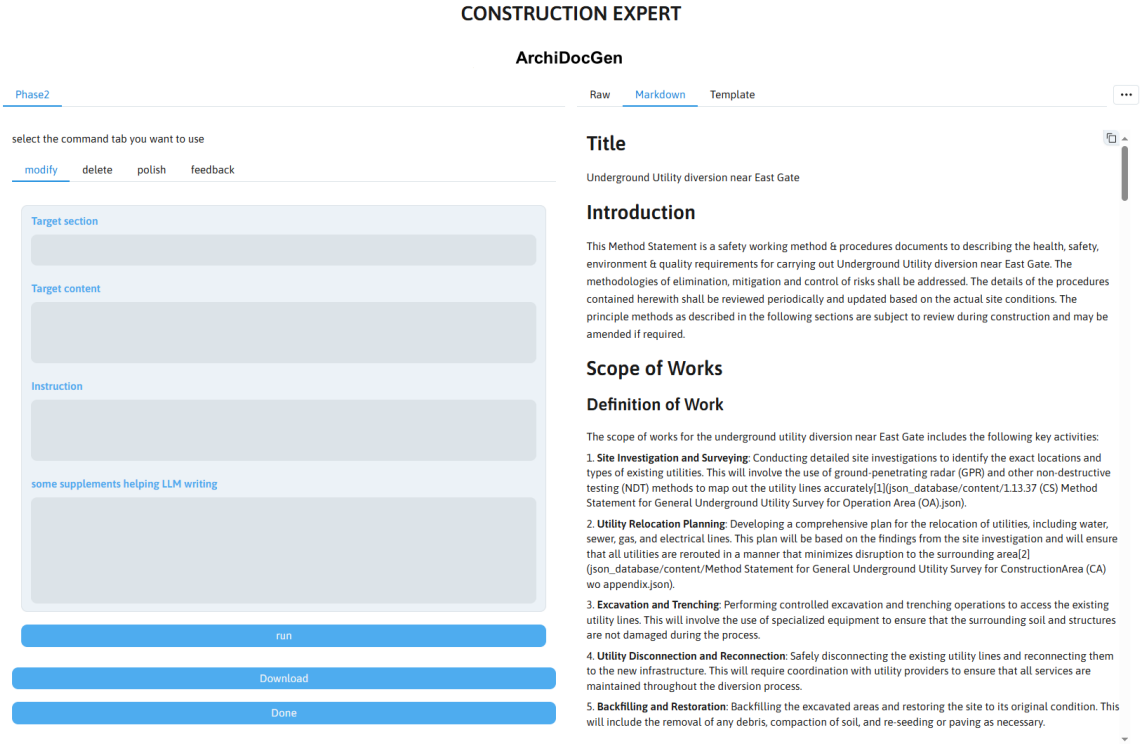


Figure 6: Screenshot of our web application used to generate method statements.

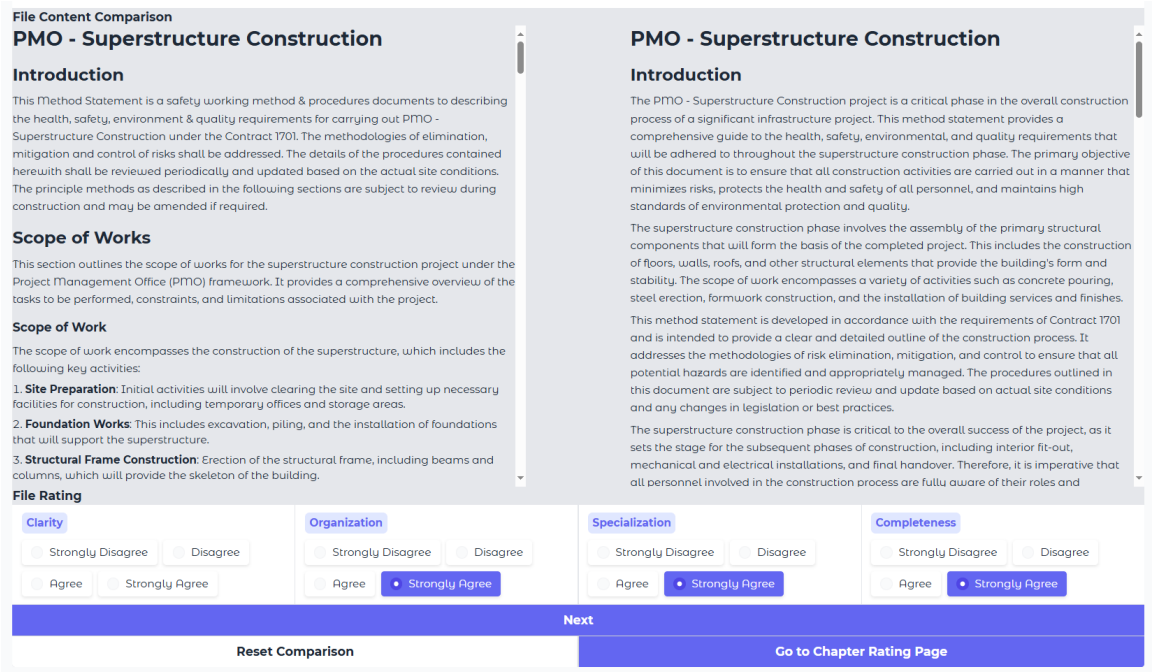


Figure 7: Comparison interface on the ArchiDocGen evaluation platform, allowing engineers to assess and rate generated method statements across four dimensions.

E Prompt Template

E.1 Template-based Generation

Template-based Generation

This method statement is a safety working method & procedures documents to describing the safety, environment & quality requirements for carrying out `{Title}`.
The methodologies ...
The content: `{Outline}`

E.2 OutlineScore Prompt

OutlineScore Prompt

Evaluate the outline below on four criteria, scoring each from 1 (lowest) to 5 (highest). Use the reference outlines as examples of top performance.

Clarity: Are main headings concise and mainly followed by secondary headings, without excessive sub-branching?
Organization: Are sections clearly distinct, with no overlaps, and do they comprehensively cover the title?
Completeness: Does the outline fully address all key points with no major gaps?
Specialization: Is the content tailored to the title, considering aspects like Work Method, Responsibility, and Quality?

Reference outline: `{criteria}`
Outline to evaluate: `{outline}`

Return only the following format:
Clarity: [Your score here]
Organization: [Your score here]
Completeness: [Your score here]
Specialization: [Your score here]

E.3 Evaluator LM Criteria

Expert-defined Criteria

Please score the construction document overall (1-5 points). Focus on the following aspects:
Completeness: Does it cover the main sections/key points, e.g., project introduction, scope of work, responsibilities, schedule and work timelines, resource requirements, construction methods, and safety/environment/quality considerations?
Clarity and Understandability: Is the information expressed clearly? Is the structure logical and easy to follow for execution and supervision?
Industry Compliance: Does it comply with basic construction standards, safety, and environmental regulations? Does it include necessary quality control measures?
Operability and Feasibility: Does the plan have practical value? Does it include executable details, timelines, and methods?
Overall Professionalism: Is the method statement detailed, logical, and capable of meeting project needs?

score1_description: The document is almost entirely useless: it is severely lacking in critical information, with no clear construction approach. There is a lack of necessary compliance or safety considerations, and overall quality is extremely poor.
score2_description: The document has some ideas, but significant gaps remain: only a few core points are covered, with many sections or critical requirements (e.g., safety, quality, environmental considerations) clearly missing. Its practicality is very low.
score3_description: The document is generally feasible: it covers the main construction points and compliance requirements but lacks depth or detail in some areas. There is some degree of practicality, but it still needs further supplements or improvements.
score4_description: The document is very close to high quality: most of the content is complete, clear, and executable. It takes safety, environmental protection, and quality management into consideration, with only minor details needing improvement.
score5_description: The document is of excellent quality: it provides a systematic and detailed description of all aspects, with full compliance with safety, environmental, and quality standards. Its practicality and clarity are excellent, meeting or exceeding industry best practices.