A USEFUL PROGRAM TO BENEFIT HUMAN WRITERS USING AI-BASED FEEDBACK AND SUPPORT

Raymond Kaige He [1] , Moddwyn Andaya [2]

[1] Issaquah High School, 700 2nd Ave SE, Issaquah, WA 98027
[2] California State University, Sacramento, 6000 Jed Smith Dr, Sacramento, CA 95819

ABSTRACT

This paper addresses the over-reliance on AI in writing, which has been shown to contribute to skill atrophy and reduced creativity [1]. Rather than attempting to replace human authors, this program aims to integrate AI and human writing in a responsible, slightly gamified writing support tool, built upon Unity and C#. The system consists of three main components - an AI controller, a task and progress checker, and a main text box where users can write and edit. Instead of generating full essays, the AI provides creative prompts which it can then check. The program encountered several challenges, including bugs, reliable AI validation, and prevention of user dependency, requiring further refinement. Experiments were conducted using twelve AI-generated essays to evaluate whether the system preferentially approves formally structured writing; acceptance rates and evaluation outcomes were analyzed to identify system behavior.

KEYWORDS

Writing Support, Human-AI Collaboration, Artificial Intelligence, Gamification, Creativity

1. INTRODUCTION

The problem in question is the overreliance on AI to write for individuals, which leads to a diminishing of true writing skill and a decline in overall proficiency. Writing is not merely a technical skill but one that develops creativity, critical thinking, and intellectual capability; therefore, replacing or heavily relying on AI for this integral skill can lead to significant negative consequences. Research on critical thinking and writing emphasizes that writing practice is essential for developing reasoning and intellectual engagement. Overreliance on AI in creative writing leads to deskilling and diminished critical engagement, resulting in a reduced ability to independently approach and develop written work. This trend becomes particularly noticeable as teachers increasingly shift back to traditional pencil-and-paper writing in response to the increasing influence of AI in education [3]. Writing development research also shows that writing proficiency emerges gradually through repeated practice and revision, reinforcing the importance of active engagement in the writing process rather than reliance on automated tools [4].

In the long term, skill atrophy and reliance on AI may cause researchers to depend on their tools rather than actively directing them. This can also lead to superficial competence, in which users may feel competent because AI-generated output looks plausible, even if users themselves don't understand the topic. Dependence also reduces originality AI-supplemented writing is mainly homogeneous and AI-created writing even more so.

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp. 65-75, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160806

AI writing tools are increasingly used by students, and research suggests that heavy reliance on such tools may affect the development of independent writing skills [5]. AI's ubiquity means that this pattern is likely to repeat itself, so it is imperative that AI should be integrated into conventional writing rather than replacing it. AI systems should focus on supporting the development of writing skills, and responsible AI use should be encouraged. Recent research on generative AI also highlights the need to recalibrate academic expertise, emphasizing that AI tools should augment human reasoning rather than replace intellectual effort [10].

Three existing methodologies were examined as comparative references for the proposed system. Methodology A is Wordcraft[11], a tool-based AI writing generator designed to support brainstorming and worldbuilding, allowing users to request different forms of AI assistance. It was effective for generating ideas and narratives but struggled to maintain authorial voice and wasn't transparent in its usage of AI generation. Its focus leaned toward creative writing rather than nonfiction. Another such program was CoAuthor, a real-time collaboration between humans and AI [12]. Writers could request solutions, then attempt to fix them. It was an attempt to emphasize working together with AI over replacement - however, AI-generated text still made up a majority of such stories. The third was an application called Read Revise Repeat, an iterative revision program. It emphasized efficiency and citation-focused feedback [13]. However, it was focused on revision rather than structured growth. The proposed application is designed to improvewriting skills by supporting users rather than replacing them, while also incorporating light gamification to encourage motivationand accommodating multiple writing genres.

The solution to this problem should be implementing responsible and practical AI use that encourages writers.The approach is intended not to detract from conventional writing and instead provides useful advice to help writers develop their skills; in addition, this also helps them through creative prompts- prompts including nonfiction, practical essay and fiction writing to encourage users to think creatively.

Additionally, this method promotes responsible integration of AI tools by teachingskill development instead of conveniently writing the user's ideas for them, preventing development of a unique voice. Unlike conventional AI writing tools that may create dependency on said tools and superficial, unviable competence, this solution maintains the user's role as the primary author and engaged in these tools. As a result, writers continue to practice essential skills such as critical thinking, originality, and revision. The tool represents a sustainable and educational application of AI that has the potential to contribute more benefits than drawbacks.

The experiment tested whether the program consistently responded positively to well-written essays. It also tested whether the AI would show bias towards text that was also generated by AI, namely itself (ChatGPT). Twelve "perfect" essays were taken from ChatGPT, Gemini and Claude, different AIs with different conversational styles [6]. Using multiple AI sources reduces the chance of self-favoritism and is more efficient than manually writing essays. It did not affect the intended purpose of the program - to help human users write - because it was a test-controlled case. Some essays were rejected due to problems involving formatting, while clearly structured and stylistically appropriate essays were accepted. In particular, the AI praised Claude more, while criticizing the writing of Gemini. Interestingly, ChatGPT rejected one of its own essays, suggesting that validity depends more on adherence to prompts and formatting than writing quality alone. The program also struggled to generate meaningful feedback for highly academic writing (such as that of Claude), implying advanced users may plateau, though developing writers would still benefit significantly.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Improving AI Output Reliability and Feedback

In order to ensure reliable accuracy, AI output could be reduced to two possible answers—true or false—from which feedback could then be generated. This approach would eliminate intermediate values that might cause the AI to fluctuate between possible solutions. In addition, users could report AI outputs they find inaccurate for further testing, which would help improve the system's accuracy in evaluating different types of writing. A failsafe mechanism could also be implemented, or users could be given a way to provide the AI with additional context about what they are writing. For example, a separate tab could be used to provide the AI with additional context through a dedicated text box, allowing users to continue developing the same piece of writing over time while improving the system's understanding of the task.

2.2. Designing AI as a Writing Support Tool

The assistant could be designed in such a way that it supports the writer rather than merely performing the writing for users. Instead of generating full paragraphs by default, the AI could provide options such as outlining, asking guiding questions (which currently lack implementation), or offering several revision strategies from which the user can choose. Training features could also be incorporated, such as gradually reducing the amount of assistance available over time (or after a certain number of prompts) or requiring users to attempt their own revision before the AI provides suggestions.

In addition, an extra reflection step could be included after feedback, such as asking the user to explain what they changed and why, so that the tool reinforces learning rather than replacing it. The system could also clearly indicate when the AI is being used and for what purpose. This approach would assist users by giving them an active voice in the writing process.

2.3. Aligning AI Feedback with User Intent

To minimize the gap between user intent and AI output, the system could include an intent selection feature prior to providing feedback. For example, users could select a specific goal, such as descriptiveness, formality, grammar, or clarity,to allow the AI to better understandthe type of assistance requested. Additionally, the system could allow users to highlight a specific section of text and choose the type of revision they want applied, rather than requiring the AI to interpret the entire draft at once.

If the AI still misunderstands the user's intent, a simple correction system could be implemented. For instance, the user could select options such as "too formal," "changed meaning," or "not what I meant," allowing the AI to adjust its approach based on that feedback. Another possible safeguard could be a "keep meaning" mode, in which the AI is required to explain its interpretation of the user's intent along with the feedback it provides, giving users the opportunity to confirm or correct that interpretation. Additionally, incentive mechanisms—such as awarding extra experience points (XP) when the AI successfully aligns with the user's intended output—could encourage user engagement with the system.

3. SOLUTION

The main structure of the program is built around three components: the textbox, the tasks controller, and the AI controller. The program follows a simple loop. First, it checks the tasks displayed to the left of the textbox. The user may then write within the textbox and choose either to receive AI assistance—such as corrections related to grammar, style, word choice, and tone—or to complete a task, which can then be checked off to earn experience points.

GPT-4o Nano was used as the AI model. The program sends a prompt to the AI and retrieves the response through the use of a shared key. The AI is instructed to format its output in a specific structure so that it can be correctly processed by the program, as it receives a preliminary prompt describing how the text should be formatted.

Tasks are randomly assigned from a relatively large list of possible options. These tasks are organized into several categories, including word-count and timer-based tasks, nonfiction writing tasks, and fiction writing tasks. Many of the nonfiction writing tasks are derived from various essay idea websites, while fiction writing tasks are largely taken from the subreddit r/WritingPrompts, which specializes in creative fiction prompts.

An extensive list of text-formatting features—such as bolding, italicization, and indentation—was initially planned but ultimately removed late in development because they proved difficult to implement within Unity. Code generation assistance was provided by GPT-5–series models, with additional user modifications and support from CodingMind. The program was written in C# and developed within the Unity game engine, while the user interface was manually designed using Unity's editor.

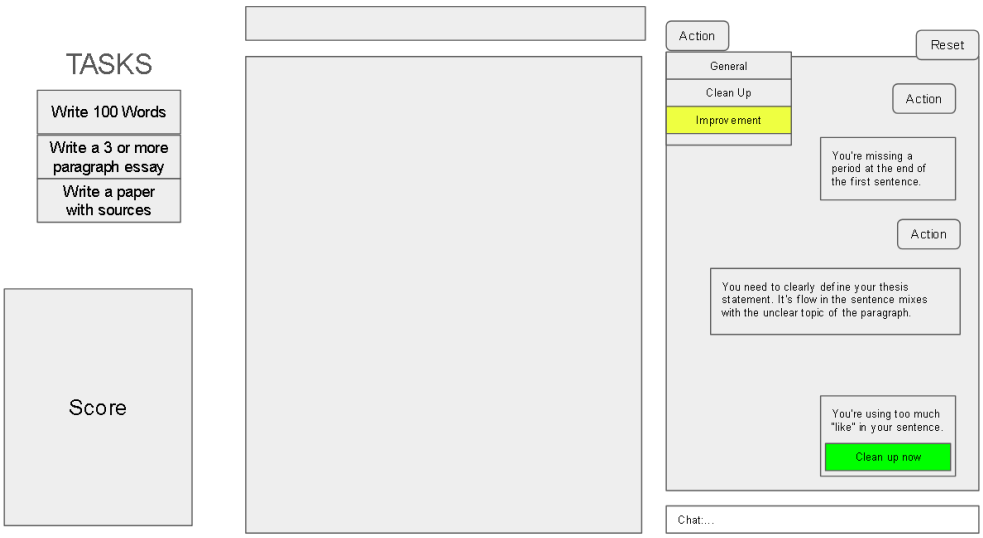

Figure 1. Overall system architecture of the AI-assisted writing support application implemented in Unity

One of the primary components of the program is the AI controller. Its purpose is to analyze user writing and provide structured feedback without generating replacement text. This component is implemented using the OpenAI API through the OpenAIClient and chat completion endpoint. It relies on Natural Language Processing (NLP), which allows machines to interpret and evaluate human language as data [7]. Within the program, this component receives user input, sends it to

the selected model, and returns formatted feedback that integrates directly with the task and XP systems.

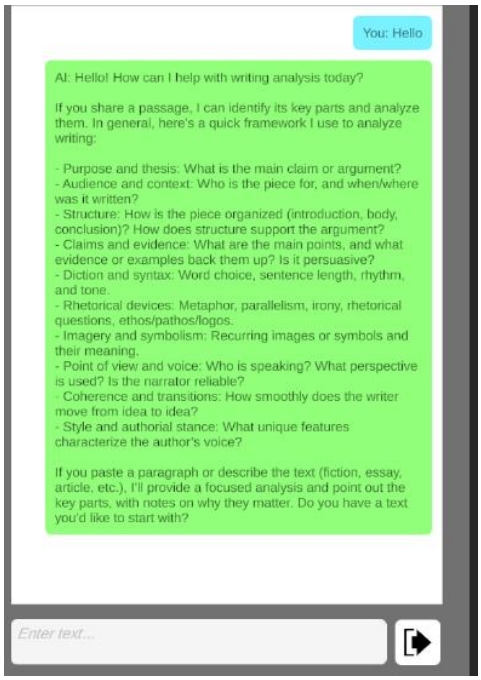

Figure 2. Workflow of the AI Controller module for analyzing user writing and generating structured feedback

```
public async Task<string> SubmitChat(string input, bool isJSON = false, bool isThinking = false)
{
    if (string.IsNullOrWhiteSpace(input)) return "";
    messages.Add(new Message(Role.User, input));

    string model = "gpt-5-nano";
    if (isThinking) model = "gpt-5";

    ChatRequest request = new ChatRequest(messages, model: model, responseFormat: isJSON ? TextResponseFormat.JsonSchema : TextResponseFormat.Text);

    try
    {
        isGenerating = true;
        IsGeneratingEvent?.Invoke(true);

        var response = await client.ChatEndpoint.GetCompletionAsync(request);
        var output = response.FirstChoice;
        messages.Add(new Message(Role.Assistant, output));
        return output;
    }
    catch (System.Exception e)
    {
        Debug.LogError("AI error: " + e);
        return "";
    }
    finally
    {
        isGenerating = false;
        IsGeneratingEvent?.Invoke(false);
    }
}
```

Figure 3. Internal processing pipeline of the AI feedback generation system using the OpenAI API

The method SubmitChat represents the core interaction between the program and the AI system. It is called whenever the user submits writing for feedback. The method begins by verifying that the input is not empty. If valid, the user's input is appended to the messages list as a Role.User message.

Next, the program selects which model to use. By default, it uses "GPT-5-nano", but if deeper reasoning is required, it switches to "GPT-5". A ChatRequest object is then constructed, containing the conversation history, the selected model, and the desired response format (either plain text or structured JSON).

Before sending the request, the Boolean is Generating is set to true, and Is Generating Event is invoked to notify the UI that generation is in progress. The program then asynchronously calls Get Completion Async. When a response is received, the first choice is extracted, stored as a Role. Assistant message, and returned.

If an error occurs, it is logged. Finally, the generating state is reset to false.

The second major component of the program is the Tasks Controller. Its purpose is to generate, track, and validate writing tasks that structure the user's progress. This component manages task instantiation, progress updates, AI validation for prompt-based tasks, and XP rewards. It functions as the gameplay and progression system of the application.

Figure 4. Architecture of the Tasks Controller module responsible for task generation, tracking, and validation

```
async System.Threading.Tasks.Task ValidateAndRewardTask(TaskSlot slot, bool manual)
{
    // Prevent double-submit for the same slot (checkbox spam / repeated clicks)
    if (slot.isSubmitting) return;
    slot.isSubmitting = true;

    bool lockTaken = false;

    try
    {
        // Queue AI calls so messages list isn't mutated concurrently inside AI wrapper
        await aiLock.WaitAsync();
        lockTaken = true;

        string prompt =
            $"Task: {slot.task.description}\n" +
            $"Text: {(textBox != null ? textBox.text : "")}";

        Debug.Log($"[TasksController] AI check start slot={slot.GetInstanceID()} type={slot.task.type}");

        string response = await taskCompletionAI.SubmitChat(prompt);

        Debug.Log($"[TasksController] AI response slot={slot.GetInstanceID()} => {response}");

        // Seperate response into lines
        if(string.IsNullOrEmpty(response))
            return;

        string[] lines = response.Split(new[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
        bool approved = lines[0].Trim().ToLower().StartsWith("true");
        string feedback = lines.Length > 1 ? lines[1].Trim() : "No feedback provided.";

        if (approved)
        {
            points += slot.task.pointValue;
            currentTaskCount++;
            AssignRandomTask(slot);
        }
        else if (manual)
        {
            if (slot.promptButton != null)
                slot.promptButton.interactable = true;

            UpdateTaskLabel(slot);
        }

        chatBox.CreateAIBubble($"Task {(approved ? "approved" : "rejected")}\nFeedback: {feedback}");
```

Figure 5. AI-based task validation and reward mechanism for evaluating prompt-based writing tasks

The method Validate And Reward Task runs when a prompt-based task requires AI validation [2]. It begins by preventing duplicate submissions through the is Submitting flag. A semaphore (aiLock) ensures that only one AI request occurs at a time, preventing concurrency issues with the shared message list.

The method constructs a prompt string combining the task description and the current contents of the text box. This prompt is sent to the AI using SubmitChat. Once a response is returned, it is

split into lines. The first line determines approval by checking whether it begins with “true.” The second line is treated as feedback.

If the task is approved, points are added and a new task is assigned. If rejected and manually submitted, the button is re-enabled. Finally, a feedback message is displayed in the chat box, and submission states are reset.

The third major component of the program is the action-button and chat interface system, implemented through ChatBoxManager. Its purpose is to give users direct, structured AI assistance options (grammar, quality, word choice, overused words, and style/tone) while also supporting normal conversation feedback. Similar chatbot-based systems have been explored in educational contexts, where conversational AI can provide immediate feedback and support language learning through interactive dialogue [9]. This component connects Unity UI buttons and input fields to two AI pipelines: an “action” AI that returns structured JSON feedback for targeted edits, and a “conversation” AI that returns general responses [8]. It relies on NLP by converting the user’s writing into prompts that the model can evaluate, then displaying the results through dynamically instantiated chat bubbles.

Figure 6. User interface of the action-button system providing structured AI assistance (Grammar, Quality, Word Choice, Overuse and Style& Tone)

```
// When we press any of the action buttons
5 references
void OnSendActionInput(string actionType)
{
    if (isAIGenerating) return;
    CreateUserBubble("You: Sending Action " + actionType);
    string fullCommand = $"{actionPrefix} {actionType} '{paperField.text}' ";
    _ = SendToAI(fullCommand, actionAI, isThinking: true, createResponseBubble: false, OnFinishResponse: (response) =>
    {
        // The response will be a JSON string, so we need to parse it
        AIActionResponse actionResponse = JsonUtility.FromJson<AIActionResponse>(response);
        CreateAIBubble("AI: " + actionResponse.output);

        string contextText = paperField.text.Substring(actionResponse.context.indexStart, actionResponse.context.indexEnd - actionResponse.context.indexStart);
        print(contextText);
    });
}
```

Figure 7. Processing pipeline of structured JSON responses for targeted writing feedback

The method On Send Action Input runs when the user presses any of the action buttons, such as Grammar or Style/Tone. It first blocks the request if an AI response is already generating. It then logs the requested action in the chat and constructs a command string using an action prefix, the selected action type, and the full writing from paper Field. text. This command is sent to the specialized action AI through Send To AI, which is called Submit Chat asynchronously.
Unlike normal conversation responses, action responses are expected to be structured data. When the response returns, the code parses the JSON using Json Utility. From Json <AI Action Response>. The parsed output field is displayed to the user as an AI chat bubble [9]. The context indices identify which part of the original writing the feedback applies to, and the code extracts that substring from paper Field .text for logging and targeting.

4. EXPERIMENT

A possible blind spot in the program is whether the AI system can reliably help users with lower skill as opposed to those with higher skill. The AI tends to return positive evaluations when high-skill users submit writing, meaning that writing progression may plateau if they use this application.

This study follows a staged evaluation approach. The current paper presents Stage 1, which focuses on testing system behavior under controlled conditions. This stage does not evaluate learning outcomes or user improvement. Later stages will incorporate human-written essays and controlled comparisons to better evaluate real-world effectiveness.

Twelve “perfect” essays generated by other AI systems were sourced, pasted into the program, and evaluated to determine whether the outputs were returned as positive or negative. The experiment was structured in this way to test whether the AI would consistently return positive results when an essay was written “well,” and whether it demonstrated bias toward writing that was also generated by AI.

The essays were sourced from ChatGPT, Claude, and Gemini, with four essays taken from each system to verify different types of input and output. This approach helped reduce potential bias toward any single system and provided a controlled method of testing without variability in human writing quality.

The data indicated that some essays were rejected for not meeting formatting rules, while those with sufficient clarity in formatting and style were accepted. The AI systems were chosen based on how formal or conversational their writing styles were. Claude performed according to expectations because its writing was more formal and academic in tone, which led the AI integrated into the program to evaluate it more positively. Meanwhile, Gemini performed less effectively in formal contexts, as it tends to adopt a more conversational and friendly tone and often communicates using lists rather than structured paragraphs.

One unexpected result was that ChatGPT rejected one of its own generated essays, labeling it invalid. This suggests that validity is not determined solely by the quality of the writing but rather by how closely the essay adheres to the given prompt. Moreover, the AI struggled to provide critical feedback on advanced academic writing, frequently offering overly positive evaluations. This observation suggests that users with stronger writing abilities may eventually reach a plateau when using the program. Nevertheless, the system may still provide benefits for less experienced writers, who are likely to gain more from the guidance it provides.

The next phase of this research will involve evaluating human-written essays to further assess the reliability and usefulness of the system in more realistic writing scenarios.

Figure 8. Experimental evaluation setup for testing AI validation behavior using AI-generated essays

5. RELATED WORK

A source that attempts to address the same problem is called Wordcraft, which is a tool-based generator similar to the proposed system and allows users to request different forms of AI-powered support for their work [11]. It was effective for brainstorming and worldbuilding, especially for fiction, though it struggled to match the writer's voice and didn't have good transparency. Unlike Wordcraft, the proposedsystemimproves this methodology by enabling the system to work for both nonfiction and fiction writing, gamifying the writing experience slightly, and trying to supplement rather than replace author's creativity.

Another similar source, attempting to tackle the same problem, is called CoAuthor [12]. CoAuthor involves a process where a personwritesand can request AI suggestions during the writing process. It was built to support cooperation between human and AI writing - the human stays in charge, much like the proposed application- and it also attempts to capture interaction data (i.e. what the writer typed vs. what the AI suggested), which the writer can accept, ignore and edit. It also gathers data to study how humans actually use these suggestions. CoAuthor is designed more as an assistant, which aims to curate and revise. However, AI-generated text still makes up a majority of the written work. In contrast to CoAuthor, the proposed application differentiates itself through its gamified structure and by avoiding AI-generated text entirely.

Another application that uses the same formula is called Read Revise Repeat (R3).It is built around iterative revision, which is close to how students actually try to improve essays [13]. The model will propose edits, then the student will accept or reject those edits (with a simple decision instead of heavy prompt-writing). It will continue doing this for the remainder of the essay.This application primarily focuses on revision quality and efficiency, with content-grounded citations, whereas the proposed system emphasizes both improvement and engagement. This distinction may suggest that the proposed system has an advantage in its ability to support structured skill development in both fiction and nonfiction writing.

6. CONCLUSIONS AND FUTURE WORK

The proposed system provides an initial platform for integrating AI with human writing in a responsible, slightly gamified writing support tool, with the potential to motivate users to engage with a variety of genres and develop their writing skills.

The results from Stage 1 of the evaluation provide preliminary insight into the behavior of the system. The system appears to respond positively to writing that follows clear structure and formatting conventions, while also showing limitations in evaluating highly advanced or academic writing. These findings should be interpreted as an initial validation of system behavior rather than a complete assessment of effectiveness. The educational impact of the system remains to be validated in future stages.

One limitation of the project is that the current task system does not consistently allow users to pursue their specific writing goals. The gamification framework is not tailored to individual development needs; instead, it relies on a set of predetermined tasks. As a result, experience points (XP) are not granted based on basic writing actions, and the introduction of a more extensive quest system could encourage users, particularly creative writers, to produce longer and more substantive essays.

Another limitation is the lack of efficiency in submitting and completing tasks, which may become problematic under strict time constraints. Additionally, there are challenges related to the design of incentives. Although an XP system is currently in place, with more development time it would have been possible to add cosmetic items, new personalities for the assistant, badges, or other engaging rewards to enhance progress and make the system more interactive and gamified. A progress-saving feature could also be implemented, allowing users to return to the application without losing accumulated XP and rewards.

Finally, the project revealed several limitations in its original scope and design. The initial objectives proved to be overly ambitious, which constrained the extent to which the final implementation achieved its intended goals. Future work will focus on refining the concept, prioritizing a smaller and more targeted set of features, and expanding the evaluation to include human participants and controlled conditions. This will allow for a more accurate assessment of writing improvement, user reliance, and overall system effectiveness.

Despite these limitations, the system represents a step toward integrating AI into writing support in a way that emphasizes skill development rather than replacement of human creativity.

REFERENCES

[1] Cardon, Peter, et al. "The challenges and opportunities of AI-assisted writing: Developing AI literacy for the AI age." Business and Professional Communication Quarterly 86.3 (2023): 257-295.
[2] Myllyaho, Lalli, et al. "Systematic literature review of validation methods for AI systems." Journal of Systems and Software 181 (2021): 111050.
[3] Bailin, Sharon, and Harvey Siegel. "Critical thinking." The Blackwell guide to the philosophy of education (2003): 181-193.
[4] Sulzby, Elizabeth. "Research directions: Transitions from emergent to conventional writing." Language arts 69.4 (1992): 290-297.
[5] Marzuki, et al. "The impact of AI writing tools on the content and organization of students' writing: EFL teachers' perspective." Cogent Education 10.2 (2023): 2236469.
[6] Welsby, Philip, and Bernard MY Cheung. "ChatGPT." Postgraduate Medical Journal 99.1176 (2023): 1047-1048.
[7] Chowdhary, KR1442. "Natural language processing." Fundamentals of artificial intelligence (2020): 603-649.
[8] Bourhis, Pierre, et al. "JSON: data model, query languages and schema specification." Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems. 2017.
[9] Haristiani, Nuria. "Artificial Intelligence (AI) chatbot as language learning medium: An inquiry." Journal of physics: conference series. Vol. 1387. No. 1. IOP publishing, 2019.

[10] Lin, Zhicheng, and Aamir Sohail. "Recalibrating academic expertise in the age of generative AI." Patterns 7.1 (2026).
[11] Lee, Mina, et al. “Wordcraft: Story writing with large language models.” Proceedings of the CHI Conference on Human Factors in Computing Systems. 2022.
[12] Lee, Mina, Percy Liang, and Qian Yang. "CoAuthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities." Proceedings of the 2022 CHI conference on human factors in computing systems. 2022.
[13] Du, Wanyu, et al. "Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision." Proceedings of the first workshop on intelligent and interactive writing assistants (in2writing 2022). 2022.
[14] Angioni, Manuela, et al. "Integrating XP project management in development environments." Journal of Systems Architecture 52.11 (2006): 619-626.