Human ratings of LLM response generation in pair-programming dialogue

Cecilia Domingo, Paul Piwek,
Michel Wermelinger and Kaustubh Adhikari
The Open University
Milton Keynes, England
cecilia.domingo-merino, paul.piwek,
michel.wermelinger, kaustubh.adhikari
(@open.ac.uk)

Svetlana Stoyanchev
and Rama Doddipatla
Toshiba Europe
Cambridge, England
svetlana.stoyanchev,
rama.doddipatla(@toshiba.eu)

Abstract

We take first steps in exploring whether Large Language Models (LLMs) can be adapted to dialogic learning practices, specifically pair programming — LLMs have primarily been implemented as programming assistants, not fully exploiting their dialogic potential. We used new dialogue data from real pair-programming interactions between students, prompting state-of-the-art LLMs to assume the role of a student, when generating a response that continues the real dialogue. We asked human annotators to rate human and AI responses on the criteria through which we operationalise the LLMs' suitability for educational dialogue: Coherence, Collaborativeness, and whether they appeared human. Results show model differences, with Llama-generated responses being rated similarly to human answers on all three criteria. Thus, for at least one of the models we investigated, the LLM utterance-level response generation appears to be suitable for pair-programming dialogue.

1 Introduction

Pair programming is a technique where two programmers work together, simultaneously, on the same piece of code. Numerous studies have reported on its benefits for students, for example improving the quality of the code they produce or increasing their confidence (Hawlitschek et al., 2023). However, the literature also highlights the challenges that hinder the wider implementation of this technique, such as scheduling issues or lack of suitable partners (*ibid.*). A solution that has been proposed is using a dialogue agent as a partner when a human partner is not available. Wizard-of-Oz studies suggest the viability of this option, as students welcomed having an AI partner (even before LLM-based dialogue systems became widely available), and they produced code of the same quality (or higher) as when pairing with a human (Kuttal et al., 2021; Robe and Kuttal, 2022). Those studies discussed the technology available at the time that could be harnessed to develop the system that they simulated, with some limitations.

In this context, the use of LLMs for response generation is promising. Firstly, they can be adapted to new domains to produce reasonable natural language responses even with little domain adaptation (Reiter, 2025, p. 11). With regards to output quality, evaluation studies suggest, for example, that LLMs can generate coherent as well as engaging stories (Seredina, 2024), whereas results are more mixed (depending partly on how well-resourced a language is) for data-to-text generation (Allen et al., 2024). Bozorgtabar et al. (2023) considered the task of feedback comment generation for writing learning (Nagata, 2019) - a specifically educational settings. Secondly, great advancements have been made recently in the automatic generation of code (Jiang et al., 2024). Nonetheless, code generation models have primarily been used as coding assistants rather than pair programmers (*ibid.*)

Some claims have been made about the use of LLMs for pair programming, but closer examination reveals that the collaborator role of a pair-programming partner is conflated with LLMs' default assistant role[1]. An exception can be found more recently in a study (Lyu et al., 2025) where different educational pair-programming settings are explored; two of those settings (a human pair with an LLM, and a solo human with an LLM) encourage the use of LLMs as collaborators. Still, the researchers found that the "LLM-based tools were primarily perceived as technical assistants" (*ibid*, p. 8), used for debugging and syntax queries. In an educational setting, these tools need to simulate a different type of role: as collaborators that engage

[1]An excellent example is a MOOC offered by Google. The title of the course suggests that LLMs may be used as pair-programming partners, but the course contents merely teach how to make API calls for code generation or code transformation, and the instructor explicitly refers to the LLMs as assistants.

Proceedings of the 18th International Natural Language Generation Conference, pages 41–59
October 29–November 2, 2025. ©2025 Association for Computational Linguistics

the learners (Grassucci et al., 2025). Thus, here we take the first steps in exploring whether LLMs are suited to step out of the code assistant role and into a programming peer role.

As a beginning step, we analyse whether they can generate single responses that are contextually appropriate in relation to the dialogue history so far, before the approach can be carried further to study whether the models' performance is consistent across a whole dialogue.

Guided by Gricean maxims of conversation (Grice, 1991) and the principles of dialogic teaching (Wegerif and Mercer, 1996), we operationalise the idea of suitability as the responses being coherent (do the responses make sense in the current context?) and collaborative (do the responses make a helpful contribution without taking over the whole task?). In line with traditions in researching dialogue systems, we also evaluate whether answers seem human to users, to assess their naturalness. These gives rise to our three research questions:

- **RQ1** Do AI responses appear human?
- **RQ2** Are AI responses as coherent as human responses?
- **RQ3** Are AI responses as collaborative as human responses?

We carry out our evaluation by asking human annotators to rate a set of responses obtained by prompting two LLMs to provide a continuation to a given dialogue context. The dialogue contexts were obtained from real pair-programming dialogues between students (Domingo et al., 2024). The raters who judged a portion of our large corpus were shown responses from our three sources (human ground truth, GPT, and Llama), without knowing the source of any of them, nor how many might be AI-generated.

Our pair-programming dialogue corpus is, as of yet, not accessible while we finalise the evaluation studies that we collected it for, to avoid data contamination (Balloccu et al., 2024). The corpus will be available to the research community for future evaluation studies, but under rigorous conditions to prevent its use as potential training material for LLMs. Nevertheless, we appreciate that over time the corpus might anyway be leaked into training data for future LLMs.

The following section summarises the main studies that have informed our work. In our methodology section, we first provide a brief description of the dataset we used. Then, we discuss how we arrived at our final prompts and model choices. Thirdly, we describe how the evaluation of the responses was carried out. Afterwards, we present and discuss our quantitative and qualitative analysis results.

2 Related work

As defined above, pair programming is a collaboration technique used in programming whereby two individuals (a pair) work on the same code simultaneously (Hanks et al., 2011). The pair can collaborate using the same computer if they are co-located, or they can collaborate remotely using the wide range of tools available for remote shared access to a programming interface (Adeliyi et al., 2021). With two people programming simultaneously, roles need to be negotiated to ensure successful collaboration; this normally results in programmers adopting the role of navigator or driver, though researchers have sometimes described pair-programming interactions through different roles (Hanks et al., 2011). The navigator contributes verbally by suggesting where the code can go, while the driver types in the code that goes in the direction agreed with the navigator. These roles may be fixed or switch through the interaction; switching may even be encouraged in educational settings for students to benefit from both roles, as in (Bigman et al., 2021)).

Pair programming has attracted much scholarly interest and its benefits and implementation have been studied in both educational and professional settings (Hawlitschek et al., 2023; Hanks et al., 2011). Among the most widely reported benefits are increased code quality and programmer confidence (Hawlitschek et al., 2023; Hanks et al., 2011; Werner et al., 2004). Despite its benefits, pair programming faces significant hurdles for its implementation in educational settings specially. Infrastructural issues can be remedied through the use of platforms and other tools for remote pair programming (Adeliyi et al., 2021; Bigman et al., 2021). However, other issues remain largely unsolved: scheduling problems and lack of suitable partners (Hanks et al., 2011; Hughes et al., 2021). One solution that has been proposed is replacing the human partner with an AI agent when no (suitable) human partner can be found. As we mentioned, this idea has been tested through Wizard-of-Oz studies (Kuttal et al., 2021; Robe and Kut-

tal, 2022). These studies showed that, even before LLM-chatbots entered everyday life, students could welcome an AI partner, and that the quality of the code produced with an AI partner could be similar or sometimes higher than when collaborating with a human partner. The authors of these studies designed the interactions based on the state of the art at the time, but since then great advances have been made in Natural Language Processing and Programming Language Processing, both separately and in conjunction (Jiang et al., 2024).

Since the release of OpenAI's ChatGPT, a large body of research has been released using this tool and other large language models — see (Liu et al., 2023) for a review focused on ChatGPT; while it seems to be the most widely used model, there are numerous open-source and open-weights alternatives (Kukreja et al., 2024). These models have surpassed previous techniques in all kinds of NLP tasks. While testing on previous NLP benchmarks can be uninformative due to possible data contamination, evidence of excellent LLM performance in many language-related tasks is abundant; see (Huzaifah et al., 2024) or (Ostyakova et al., 2023) for some interesting examples among the many available. Moreover, they are making advanced NLP tools increasingly accessible, as often good results can be obtained without even the need to fine-tune (Liu et al., 2024), and even zero-shot use can be sufficient in some scenarios through effective prompting (White et al., 2023). Additionally, newer large language models are being released with an emphasis on efficiency, making their use feasible even without GPUs (e.g., high-performing small variants of the Llama family can be run on simple CPUs[2]). This growing smorgasbord of LLMs also features different types of models with regard to their openness (ranging from commercial models to fully open models), further increasing their accessibility (Jiang et al., 2024). LLMs are not only state-of-the-art models of popular natural languages — their training data also includes code. While performance is not always up to par (Wermelinger, 2023), LLMs have generally demonstrated good performance at tasks involving code (Austin et al., 2021; Coignion et al., 2024). Thus, in the current landscape, it appears that LLMs could be ideally suited to fulfill the role envisioned in the cited Wizard-of-Oz studies (Kuttal et al., 2021; Robe and Kuttal, 2022): having AI as a pair-programming partner when no suitable human partner is available. In addition to large size, one important ingredient for LLMs' success is instruction tuning (Jiang et al., 2024). This optimises the models for dialogue use and task completion via prompting, beyond the simple objective of text completion.

LLMs are already revolutionising the education sector, for better or worse. They have been used in many instances for Computing Education, primarily for question-answering and debugging (Ferino et al., 2025). However, to harness the educational potential of these tools, students need to be engaged as active agents in their learning, rather than passive recipients of information (Grassucci et al., 2025). For that to happen, models need to be guided away from a helpful assistant role into an engaging collaborator role, be it through few-shot learning or finetuning (Yuan et al., 2025). That switch in roles could allow LLMs to be used in dialogic teaching practices, which require collaboration (Wegerif and Mercer, 1996). LLMs have already shown their ability to imitate students in some contexts (Ma et al., 2024) — can they also act as pair-programming partners?

3 Methodology

3.1 Our dataset

Even though pair programming is a widely studied topic, research data is not so widely available: studies often focus on the product of the interaction (e.g., the code produced, and course assessments and retention rates in educational settings), or on settings where it might be challenging to release data – e.g., private companies (Plonka et al., 2015) – or are only able to release limited text data (Robe et al., 2020). However, pair-programming dialogue is inherently multimodal: if we adopt Clark's conceptualisation of dialogues as highly linguistic activities in the broader spectrum of joint activities (Clark, 2005), contributions to the code are a key non-verbal element of the joint activity.

We collected a multimodal pair-programming dataset[3] of 25 dialogues between students from our institution, thus focusing on an educational setting. For practical purposes, we opted for remote sessions. For this study, the data types that we used from our dataset were the session transcripts[4], and

[2] https://llamaimodel.com/requirements-3-2/

[3] Due to ongoing anonymisation efforts and concerns about data contamination, at the time of writing we are only able to release one json file. The complete dataset, including video and audio recordings, will be released in January 2026.

[4] Recordings were pre-transcribed using Whisper (Radford

Python code files.[5]

3.2 Prompt engineering and model selection

We worked with the GPT and Llama LLM families, as two representative model families of closed and open-weights models, respectively (Jiang et al., 2024). We select two models in our final evaluation to observe possible model differences, as well as to provide insights on both commercial models widely used in research and a more accessible open-weights alternative. We use one of the dialogues as our development set to test our prompts and make the final model selection. Since our data includes code, we started our tests with the CodeLlama series of models (based on Llama 2), moving on to newer models. As can be seen in our Supplementary Materials (Part C), our prompts included instructions to the model, some dialogue history, and, in later tests, a few-shot example. In our instructions, we transitioned from a focus on the characteristics of the expected response to a focus on the persona to be adopted by the model; that proved more effective at achieving the desired characteristics. With regard to formatting, we formatted the dialogue history as json for improved input processing, even when we did not request json output.[6] We tested different context-window lengths, from 4 to 50 turns. We also tested different types and lengths of few-shot examples: from 27 to 50 turns, from both a real and a fabricated dialogue sample. With the real sample, we also tested extracting the sample from different points of the dialogue.

For our preliminary evaluation, we obtained over 100 samples per model and prompt combination tested (the number varied due to varying context-window sizes). We then carried out a simple qualitative analysis of (∼10%) randomly chosen outputs, looking at whether the model followed instructions. This allowed us to refine our prompt and decide on the optimal context. The final prompt is shown in Table 1. We also performed a quantitative analysis looking at word count, presence of formatting labels, code length, and relevant expression types (expressing thoughts or uncertainty, and expressions characteristic of an overly helpful assistant, such as "Here's the code"). We also performed this analysis on a set of 5 human dialogues to have a baseline. Our analysis showed that human responses are less verbose than model responses; using a persona-focused prompt was the most important factor for reducing the model's verbosity closer to a human level. We also observed that the "eager-assistant" style of expressions was more common in CodeLlama models, which motivated us to disregard these models. Lastly, we saw that all Llama models readily imitated the style of the dialogue context with expressions typical of human speech (e.g., hesitations like "uh"). However, these were produced in excess until the prompt was refined.

Further details from the preliminary quantitative analysis can be found in the Supplementary materials (Part A). Based on our preliminary analyses, for our final analysis we opted for a persona-focused prompt with json format and a 50-turn dialogue context and a real 50-turn few-shot example. From the Llama family, we selected Llama 3.2 1B, as it coupled efficiency and good performance. Going from the small CodeLlama 7B to the 13B version roughly doubled inference time; meanwhile, inference times with the more recent and smaller Llama 3.2 were about half those of CodeLlama 7B, with better responses as well. From the GPT family, especially as they are commercial models, we also opted for an efficient, and thus cost-effective, model; we chose GPT4o-mini, whose performance is not far from the full model in several benchmarks[7].

3.3 Human evaluation

Seeing that longer contexts yielded better results, we set the context length to 50 turns and we extracted query points for all our remaining dialogues. The query points were to be used as input for the model, containing the system prompt with instructions, dialogue history (50 turns), and few-shot example (50 turns), and the user prompt as the last utterance after the dialogue history, the utterance that the model had to respond to. We then extracted a random evaluation split consisting of 10% of query points. The splitting was done balancing the following features: length of the ground-truth response, whether the ground-truth response changed the code, and the position in the dialogue (beginning, middle, or end — as our dialogue history length was set to 50, no query point could come

et al., 2022) and revised manually. Diarisation was carried out using the pyannote library and revised manually

[5]Code was recorded every time a change was made.

[6]For our application, json provided the most convenient format - regardless, it seems that the specific choice of data representation may not have a large impact on neural generation results (e.g., Howcroft et al., 2024).

[7]https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context. Description of CODE STATE label. Task instructions separated from dialogue context. Few-shot example included in json format.

PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
Below is an example excerpt from conversation between two students in the same setting, solving a different task:

EXAMPLE TASK: <FEW-SHOT EXAMPLE>
As in the example, respond following the context below. You see that the code appears after the CODE STATE tag; you can see that there's no need to preamble it, you and your partner are both aware of that tag.
You can also see that, if there are no changes in the code, it is the same as in the previous turn.
Before seeing the dialogue context, here is the task instructions, formatted as comments:
<TASK INSTRUCTIONS>
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Table 1: Prompt description and example structure for pair programming scenario.

from the absolute beginning, but rather 50 turns from the start, which is equivalent to a duration of ∼3 minutes). This resulted in 294 samples, from which a random selection of 62 was used for our current study. The average utterance length in the set was 41 characters (SD = 53), compared to the overall average of 43 characters (SD = 59). The code was changed 16.3% of the time in the set, and 17.5% in the overall dataset.

The selected models (GPT4o-mini and Llama 3.2 1B) were prompted on the evaluation query points. The model responses and human ground truth (see Supplementary Materials, Part D for examples) were then integrated into Excel files to be evaluated by human raters. For each query point, a file was generated containing: the instructions for the programming task, a summary of the dialogue context (generated by GPT4o-mini after we tested it returned accurate summaries), a reduced dialogue history of 10 turns with the code shown as images for easier visualisation, and a sheet with the three responses (human ground truth, Llama, GPT) in random order, alongside the rating menu. The raters did not know how many answers might be AI-generated. The context summary was included because we could not expect human raters to read as many turns of dialogue history as the model. The rating criteria were selected as an operationalisation of the Gricean maxims of conversation (Grice, 1991) and the principles of dialogic teaching (Wegerif and Mercer, 1996) in a way that could be accessible to raters and balance providing us with enough information and overwhelming raters. Our main rating criteria thus were:

- Coherence: Does the answer make sense in the context?
- Collaborativeness: Does the answer contribute to the task while not taking over it?

Following research traditions in dialogue systems, we complemented our task-oriented metrics by asking raters to assess whether the responses seemed human or AI-generated. While, as demonstrated by the cited Wizard-of-Oz studies (Kuttal et al., 2021; Robe and Kuttal, 2022), students would welcome an AI partner, it is still relevant to see whether models respond in a human-like manner, if we take human-style communication to be the ideal standard for communication with human users. This rating category was given binary labels (human, AI), as was the Coherence label (coherent, not co-

Criterion	Values
Coherence	- Coherent - NOT coherent
Collaborativeness	- Collaborative - Neutral - NOT collaborative
Humanness	- Human - AI

Table 2: Rating criteria

herent); we strove to make the rating as straightforward as possible for the annotators in this manner. The Collaborativeness category, however, required three levels (collaborative, neutral, not collaborative), as there were many contributions where the speaker simply showed agreement, making a neutral contribution to the task. This division into three categories instead of three was also motivated by the distinction of more than one way to collaborate in the educational dialogue literature (Wegerif and Mercer, 1996), where cumulative talk is recognised as valuable, instead of only the more task-advancing exploratory talk.

The evaluation was carried out with 16 raters; 11 were computing PhD students, while the other 5 were staff and PhD students from other departments who had demonstrated a knowledge of Python through earlier collaboration. Of the raters, 4 had previously participated in the data collection; even though the dialogue data was anonymised, we checked the anonymisation records to ensure that raters did not see any dialogue they had participated in. Given the complexity of the task, we could only use raters whom we knew had some basic programming knowledge and could be expected to show good work ethics — these criteria overlap with those for participation in the initial data collection, which is why there was some overlap in participants at both stages. The rating criteria and procedure were explained in writing and live; where raters did not attend the live session, feedback from the live session was used to improve instructions for the asynchronous raters. The raters, as were the dialogue participants, were 25% female (not self-reported, based on personal knowledge). All participants received compensation. Raters worked at varying paces; we observed some needing 5 minutes per file, while others needed 15. As the task thus demanded a lot of time from raters, and we needed samples to be rated by more than one person due to the potential subjectivity, we only had 186 query points evaluated (each containing three responses, as mentioned early). This resulted in a total of 585 ratings (each rating consisting of three sub-ratings, one for each of our criteria: Coherence, Collaborativeness, Humanness).

3.4 Analysis

We first performed a descriptive quantitative analysis to observe which features seem to play an important role in our data; we then used those insights to carry out our inferential analysis. Both types of analysis expanded upon the features that we considered in our preliminary analysis (features 1-4):

1. Presence of LLM-style phrases (in our case, variations of "Here's the code/solution").

2. Presence of markers of spoken language (filler sounds like "uh", etc., or sentences starting with "So,".

3. Response length, measured in characters.

4. Presence of CODE STATE label. This label was used to format the code in the responses and the dialogue history. We introduced this label in all human responses and instructed the LLMs to use it, but the LLMs did not always do that.

5. Response similarity with last turn. We measured sentence similarity using Google's Universal Sentence Encoder via Martino Mensio's wrapper.

6. Response similarity with the human ground truth. Naturally, the value was 1 (maximum similarity) for human responses, so these were excluded from the model that tested this feature.

7. Similarity of the response code with the code in the last turn. We measured similarity with the difflib Python library.

8. Whether the rater judged correctly the source of the response (human or AI).

The inferential analysis, following our research questions, relied on separate regression models for each of our response variables: coherence, collaborativeness, and humanness. While the human

annotators rated each variable separately, their judgments might be related, so we included the response variables as predictors as well, together with the relevant features from our list above. The feature selection was informed by the descriptive statistics: we limited the models to the predictor variables for which a possible effect was noticeable, to avoid convergence issues. As we performed multiple tests, we adjusted our alpha to $p < 1.8e^{-2}$ using Bonferroni correction. We included the annotator and the sample ID as random effects due to their high variability. All responses were rated by at least two people, but some were rated by as many as eleven to be able to calculate inter-rater agreement. Cohen's kappa is 0.04 for coherence, 0.19 for collaborativeness, and 0.52 for humanness, indicating high subjectivity in the ratings. More details on the inferential analysis can be found in the Supplementary Materials (Part B).

The quantitative analyses were complemented by two qualitative analyses: an analysis of the human-ground-truth responses that were rated as seeming AI-generated, and an analysis of raters' comments on what made them label responses as human or AI.

4 Results and Discussion

Table 3 shows the relative frequency of each rating for each of the response sources (human, Llama, or GPT). Results marked with an asterisk (*) are supported as statistically significant in our inferential analyses ($p < 1.8e^{-3}$).

	Human	Llama	GPT
Human-like	84.6*	74.4*	20.1*
Coherent	65.6	73.8	70.8
Collaborative	35.1	43.1	52.1
Neutral	50.0	41.5	10.8
NOT collaborative	14.9	15.4	37.1

Table 3: Response ratings by source, as percentage. The figures are relative frequencies: (number of ratings for this category value/total ratings in this category) × 100. Coherence and humanness are binary criteria, so only the positive category is shown. Asterisk * denotes statistical significance.

4.1 RQ1: Do AI responses appear human?

4.1.1 Results

To answer whether LLMs can return human-like responses in pair-programming dialogue, we must make a distinction between models, as the source of the response (human ground truth, Llama, or GPT) had a significant impact on humanness ratings. We see that human responses were rarely not rated as human-like, though some AI responses were rated as human-like, especially those from Llama. Table 4 shows the accuracy of annotators' judgments, with an average of 64.73%, maximum of 74.07%, and minimum of 45.24%. There were 30 instances (15.38% of human responses) where a human response was rated as seeming AI-generated. We analysed what might have caused this, and saw that half those instances were due to a single annotator being misled by the formatting of the responses (the CODE STATE label shown in all human responses, 95.38% of Llama responses, and 37.95% of GPT responses); this annotator judged all their human samples as AI, but still got 60% accuracy by applying this same judgment to AI responses containing the label. Of the remaining instances, half were instances of the same response misleading several raters: "So user"[8]. Four other instances come from the same two annotators, raters 116 and 109.

Correct judgments	Raters (ID numbers)
70-75%	101, 106, 110
65-70%	104, 105, 108, 111, 112, 113, 115
60-65%	102, 103, 109, 114
55-60%	NA
50-55%	107
45-50%	116

Table 4: Annotators' percentage of correct judgments about whether the answers were human or AI

Humanness ratings were also correlated with the similarity of the AI response with the human ground truth ($p= 5.26e^{-9}$). [9]. When an AI answer is totally different from the GT (0), its probability of being rated as human is 0.92% (i.e., almost 0). When it has a similarity of 1 (it is the same as the ground truth), its probability of being rated as human is 87.26%. The mean similarity is 0.638109, for which the probability of being rated as human is 38.60%[10].

[8] Here the speaker referred back to the "user" variable that they were defining to implement a game script, though their partner had moved one to a different part of the code.

[9] See Supplementary Materials (Part B) for more details about the similarity measures.

[10] Inferential probabilities extracted from our regression models.

4.1.2 Discussion

Even though both LLMs received the same input, Llama responses were judged as human-like much more often than GPT responses. Raters accurately judged human answers as human and often believed (less accurately) that Llama answers were human; this highlights the model's ability to imitate the style of the speakers from the dialogue context. Imitating the style of the input also moves the responses away from the default style associated with LLMs. This LLM style seems to be very present in users' minds: when we asked raters to comment on what led them to think a response was human or AI-generated, most comments tended to focus on clues pointing to AI text. When raters did comment on features that made a response appear human, they mentioned markers of spoken language (primarily "uh" and "um"). This is one way in which responses showed what raters identified as clear signs of human speech: signs of hesitation in speaking, signalling a train of thought. One annotator, however, showed awareness of how LLMs can mimic those imperfections of human speech: "when AI was not [sic] being collaborative it would be harder to notice, as humans may also respond things like 'Ah' (as it seems from the recordings). This part seemed to me harder to tell if it was human/AI generated".

Four annotators mentioned that long responses seemed AI-generated, though response length was not a significant variable. Answers rated as AI have an average of 75.48 characters, whereas answers rated as human have an average of 75.13 characters. The difference was thus minimal, and the range of response lengths was the same for both ratings. The difference is similar when we look at the actual source of the answers: 75.14 characters for human answers, the same for Llama answers, and GPT 75.48 for GPT answers. Related to this, another factor mentioned by four annotators that made them think an answer was AI-generated was an abundance of details and explanations. As there was no noticeable length difference, perhaps this has more to do with information density and perceived length, rather than objective length.

The influence of the similarity with the human ground truth was a surprising finding. While similarity-based metrics can be useful in domains like machine translation, their use in dialogue response evaluation is criticised, as valid outputs in dialogue may be more diverse (Liu et al., 2017). Nonetheless, these options may be constricted in task-oriented dialogues, as our results suggest — it is for instance quite remarkable that the average similarity is as high as 0.6 (when considering only responses whose length is similar to the ground truth, this average goes up to 0.85 for Llama responses, but stays at 0.6 for GPT responses). That being said, we must note that seeming human-like is only one aspect of the responses, and similarity with the ground truth is not related to the other suitability criteria.

4.2 RQ2: Are AI responses as coherent as human responses?

4.2.1 Results

Results concerning Coherence ratings showed fewer features having a significant effect under the Bonferroni-adjusted alpha, probably due to high variability in annotators' ratings (accounted for in our regression models). However, we did observe a significant effect of Collaborativeness ratings on Coherence ratings ($p = 7.36e^{-5}$): when utterances were rated collaborative, they were more likely to be rated coherent.

4.2.2 Discussion

Even though no other features besides collaborativeness ratings were statistically significant in relation to coherence, as we see on Table 3, our descriptive statistics show, overall, that responses were deemed coherent most of the time. The responses' internal Coherence is no surprise given how LLMs are trained on vast amounts of language data; Coherence with the broader dialogue context shows that the models are able to utilise the dialogue context effectively.

4.3 RQ3: Are AI responses as collaborative as human responses

4.3.1 Results

As with RQ2, we did not observe many significant features. The only significant result was the effect of Coherence on Collaborativeness ($p = 6.28e^{-4}$).

4.3.2 Discussion

As results regarding this criterion were mostly not significant, we cannot draw any clear conclusions. Nonetheless, if we look at Table 3, we can see a clear difference between the LLM responses and the human responses. Responses from GPT are the ones most often rated as collaborative, but also the ones most often rated as NOT collaborative;

they're rarely rated as neutral. The responses of Llama, on the other hand, are distributed across the Collaborativeness values in a way more similar to the human responses. In fact, we see that Llama responses are rated as collaborative slightly more often than human responses, while being rated as NOT collaborative at an almost equal rate to the human answers. The difference between the models with regard to Collaborativeness can be attributed primarily to Llama's ability to replicate the style of the dialogue context, where many responses were simple signs of agreement in the form of "Yeah" — shows of (dis)agreement were considered neutral in terms of Collaborativeness, signifying a minimal advancement in the task; following Wegerif and Mercer's (1996) taxonomy of classroom collaborative talk, this cumulative talk – which adds uncritically to what has gone before, occasionally with superficial amendments – is also valuable for collaboration.

When asked about what made them think an answer was human or AI, one third of raters identified some specific phrases characteristic of LLMs. One annotator actually pointed out the same phrases that we detected in one of our features (see Supplementary Materials (Part B): "In my experience, AI typically says 'Here's the corrected code:'/ 'here's how we can. . . ' so when I see that I would think it's AI-generated". Such "LLM-style" expressions could have an effect on Collaborativeness, as they introduce a solution and reflect instruction-tuned models' solution-oriented design. Raters also commented on this: "I remember that the most AI generated looking answers were the least collaborative as it would go and solve the problem itself". However, our limited, highly variable data, as well as our conservative detection of LLM-style phrases, prevented us from confirming the observations of our qualitative analysis quantiatively.

5 Conclusions

We set out to explore whether recent LLMs have brought the state of the art to a stage where an AI pair-programming partner could be easily developed. We tested Llama 3.2 1B and GPT 4o mini on Humanness, Coherence, and Collaborativeness. Both models returned good responses in terms of Coherence, though for our other suitability measures Llama showed much better performance than GPT. Its ability to imitate the style of the input dialogue made its answers seem more human-like. As a side effect, Llama responses were also rated as more collaborative overall, since they included some instances of cumulative talk and were rarely rated as not collaborative. Responses from GPT often showed a style easily recognised as characteristic of LLMs; this style also led GPT responses to often be rated as not collaborative, as the prototypical LLM-style response features a complete solution, instead of making the user a participant in the development of the solution.

Llama's already promising performance was obtained through few-shot learning, showing a promising path where large datasets might not be needed to adapt a general-knowledge model to our task. Nonetheless, we will release a modest dialogue dataset which could be utilised to explore more data-intensive approaches (e.g., fine-tuning pre-trained models). Future work also needs to confirm whether performance remains consistent through a whole dialogue. Llama showed good performance at the utterance level through different stages of the evaluation dialogues; however, performance consistency has not yet been tested in a more realistic scenario of live interaction.

What seems clear is that, although models have been trained to be helpful, this does not always translate into them being collaborative. In settings where what is helpful differs from obsequious subservience, it is thus necessary to teach the models a different attitude. Fortunately, as we have shown, with certain models, a few-shot approach may be sufficient.

Limitations

While we managed to obtain 585 rating samples, the high subjectivity of the task prevented us from obtaining many statistically significant results. An even larger sample size might have allowed us to confirm the results that appeared relevant (noticeable differences in descriptive statistics, moderate effect sizes, and raters' comments) but were not statistically significant under a Bonferroni-adjusted alpha. The subjectivity of the task also limited the number of unique query points that could be annotated (186), as we needed each point to be rated by more than one annotator. The complexity of the task meant that annotators were not easily recruited: the task was time consuming and required some knowledge of programming and analytical skills.

Another limitation of our study is the fact that we only compared two models. Naturally, it would

not be feasible to test the large number of LLMs that exist nowadays, so we selected two recent models from two representative families of closed and open-weights models (Jiang et al., 2024). Their widespread use makes it easier to compare insights from other studies. Moreover, we used small, accessible models, which would enable other researchers to replicate and expand our methodology.

We worked with a fixed context window for better comparability between responses. One drawback of this approach is that we could not query the models on utterances before turn 50, where our minimal context window would end. Based on our results, we could expect the GPT model's responses to be unaffected. The Llama model proved more sensitive to the dialogue history; however, our relatively lengthy few-shot example of 50 turns might provide sufficient context for the Llama model to maintain the style that we have observed in later turns.

While we wanted to explore how LLMs could perform as pair-programming partners, time constraints allowed us to only take an initial step. Results at the utterance level are promising, but more work is needed to confirm whether performance is consistent through whole dialogues before further steps can be taken towards the development of a full system. A dialogue-level evaluation would also require the evaluation of further variables beyond the discourse, looking also at code problem-solving skills and the learning gains from interacting with the system.

Ethical considerations

By having human annotators rate our evaluation data, we have also been able to make some observations about how users perceive AI. With the current omnipresence of LLMs, users, at least those in academic settings, have developed good awareness of the characteristics of LLM-generated output. Still, even knowledgeable users can be misled when models mimic the characteristics of natural speech. This makes it increasingly important for AI research to always bear in mind ethical considerations and disclose AI use. We hope to see students benefiting from an AI pair-programming partner in the near future, one that is as good as a human partner, but students and instructors should always be aware that they are using AI. Moreover, while an AI partner would make pair programming more accessible, efforts should still be made through other means to bring about the social benefits that AI could never fully bring.

This research project has been reviewed by, and received a favourable opinion from, The Open University's Human Research Ethics Committee. Participants gave informed consent for the use of their data, which has been anonymised. They were informed of their right to withdraw from the study, which nobody exercised after participation. While participation was voluntary, participants received a voucher as a token of gratitude.

Acknowledgments

This work has financial support from EPSRC Training Grant DTP 2020-2021 Open University and Toshiba Europe Limited. We thank our annotators for their valuable contribution.

References

Adeola Adeliyi, Michel Wermelinger, Karen Kear, and Jon Rosewell. 2021. Investigating Remote Pair Programming In Part-Time Distance Education. In *United Kingdom and Ireland Computing Education Research conference.*, pages 1–7, Glasgow United Kingdom. ACM.

Alyssa Allen, Ashley Lewis, Yi-Chien Lin, Tomiris Kaumenova, and Michael White. 2024. OSU CompLing at the GEM'24 data-to-text task. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 100–111, Tokyo, Japan. Association for Computational Linguistics.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program Synthesis with Large Language Models. *arXiv:2108.07732 [cs]*. ArXiv: 2108.07732.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta. Association for Computational Linguistics.

Maxwell Bigman, Ethan Roy, Jorge Garcia, Miroslav Suzara, Kaili Wang, and Chris Piech. 2021. PearProgram: A More Fruitful Approach to Pair Programming. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 900–906, Virtual Event USA. ACM.

Behzad Bozorgtabar, Dwarikanath Mahapatra, and Jean-Philippe Thiran. 2023. Amae: Adaptation of pre-trained masked autoencoder for dual-distribution anomaly detection in chest x-rays. *Preprint*, arXiv:2307.12721.

Herbert H. Clark. 2005. *Using language*, 6. print edition. Cambridge University Press, Cambridge.

Tristan Coignion, Clément Quinton, and Romain Rouvoy. 2024. A Performance Study of LLM-Generated Code on Leetcode. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pages 79–89, Salerno Italy. ACM.

Cecilia Domingo, Paul Piwek, Michel Wermelinger, and Svetlana Stoyanchev. 2024. Annotation Needs for Referring Expressions in Pair-Programming Dialogue. In *Proceedings of the 28th Workshop on the Semantics and Pragmatics of Dialogue - Poster Abstracts*, Trento, Italy. SEMDIAL.

Samuel Ferino, Rashina Hoda, John Grundy, and Christoph Treude. 2025. Junior Software Developers' Perspectives on Adopting LLMs for Software Engineering: a Systematic Literature Review. *arXiv preprint*. ArXiv:2503.07556 [cs].

Eleonora Grassucci, Gualtiero Grassucci, Aurelio Uncini, and Danilo Comminiello. 2025. Beyond Answers: How LLMs Can Pursue Strategic Thinking in Education. *arXiv preprint*. ArXiv:2504.04815 [cs].

Paul Grice. 1991. *Studies in the Way of Words.* Harvard University Press.

Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, and Carol Zander. 2011. Pair programming in education: a literature review. *Computer Science Education*, 21(2):135–173.

Anja Hawlitschek, Sarah Berndt, and Sandra Schulz. 2023. Empirical research on pair programming in higher education: a literature review. *Computer Science Education*, 33(3):400–428.

David M. Howcroft, Lewis N. Watson, Olesia Nedopas, and Dimitra Gkatzia. 2024. Exploring the impact of data representation on neural data-to-text generation. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 243–253, Tokyo, Japan. Association for Computational Linguistics.

Janet Hughes, Karen Kear, Bobby Law, Brendan Murphy, Jon Rosewell, Ann Walshe, Michel Wermelinger, and Adeola Adeliyi. 2021. Remote Pair Programming. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 1289–1289, Virtual Event USA. ACM.

Muhammad Huzaifah, Weihua Zheng, Nattapol Chanpaisit, and Kui Wu. 2024. Evaluating Code-Switching Translation with Large Language Models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6381–6394, Torino, Italia. ELRA and ICCL.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A Survey on Large Language Models for Code Generation. *arXiv preprint*. ArXiv:2406.00515 [cs].

Sanjay Kukreja, Tarun Kumar, Amit Purohit, Abhijit Dasgupta, and Debashis Guha. 2024. A Literature Survey on Open Source Large Language Models. In *Proceedings of the 2024 7th International Conference on Computers in Management and Business*, pages 133–143, Singapore Singapore. ACM.

Sandeep Kaur Kuttal, Bali Ong, Kate Kwasny, and Peter Robe. 2021. Trade-offs for Substituting a Human with an Agent in a Pair Programming Context: The Good, the Bad, and the Ugly. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–20, Yokohama Japan. ACM.

Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2017. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. *arXiv preprint*. ArXiv:1603.08023 [cs].

Xukai Liu, Ye Liu, Kai Zhang, Kehang Wang, Qi Liu, and Enhong Chen. 2024. OneNet: A Fine-Tuning Free Framework for Few-Shot Entity Linking via Large Language Model Prompting. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13634–13651, Miami, Florida, USA. Association for Computational Linguistics.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. *Meta-Radiology*, 1(2).

Wenhan Lyu, Yimeng Wang, Yifan Sun, and Yixuan Zhang. 2025. Will Your Next Pair Programming Partner Be Human? An Empirical Evaluation of Generative AI as a Collaborative Teammate in a Semester-Long Classroom Setting. ArXiv:2505.08119 [cs].

Yiping Ma, Shiyu Hu, Xuchen Li, Yipei Wang, Shiqing Liu, and Kang Hao Cheong. 2024. Students Rather Than Experts: A New AI For Education Pipeline To Model More Human-Like And Personalised Early Adolescences. *arXiv preprint*. ArXiv:2410.15701 [cs].

Ryo Nagata. 2019. Toward a task of feedback comment generation for writing learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3206–3215, Hong Kong, China. Association for Computational Linguistics.

Lidiia Ostyakova, Veronika Smilga, Kseniia Petukhova, Maria Molchanova, and Daniel Kornev. 2023. ChatGPT vs. Crowdsourcing vs. Experts: Annotating Open-Domain Conversations with Speech Functions. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Prague, Czech Republic.

Laura Plonka, Helen Sharp, Janet van der Linden, and Yvonne Dittrich. 2015. Knowledge transfer in pair programming: An in-depth analysis. *International Journal of Human-Computer Studies*, 73:66–78.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv preprint*. ArXiv:2212.04356 [cs, eess].

Ehud Reiter. 2025. *Natural Language Generation*. Springer Nature Switzerland.

Peter Robe, Sandeep Kaur Kuttal, Yunfeng Zhang, and Rachel Bellamy. 2020. Can Machine Learning Facilitate Remote Pair Programming? Challenges, Insights & Implications. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–11, Dunedin, New Zealand. IEEE.

Peter Robe and Sandeep Kaur Kuttal. 2022. Designing PairBuddy—A Conversational Agent for Pair Programming. *ACM Transactions on Computer-Human Interaction*, 29(4):1–44.

Daria Seredina. 2024. A report on LSG 2024: LLM fine-tuning for fictional stories generation. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 123–127, Tokyo, Japan. Association for Computational Linguistics.

Rupert Wegerif and Neil Mercer. 1996. Computers and Reasoning Through Talk in the Classroom. *Language and Education*, 10(1):47–64.

Michel Wermelinger. 2023. Using GitHub Copilot to Solve Simple Programming Problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 172–178, Toronto ON Canada. ACM.

Linda Werner, Brian Hanks, and Charlie McDowell. 2004. Pair-Programming Helps Female Computer Science Students. *ACM Journal of Educational Resources in ComputingACM Journal of Educational Resources in Computing*, 4(1):1–8.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *arXiv preprint*. ArXiv:2302.11382 [cs].

Shuzhou Yuan, William LaCroix, Hardik Ghoshal, Ercong Nie, and Michael Färber. 2025. CoDAE: Adapting Large Language Models for Education via Chain-of-Thought Data Augmentation. _eprint: 2508.08386.

A Supplementary material: Preliminary quantitative analysis for model selection and context engineering

Table 5 the results from our preliminary quantitative analysis, which allowed us to select our prompting approach. The figures for each approach are the average for all responses to the query points obtained from our development set containing one dialogue (the number of query points varied depending on the dialogue history length selected; the dialogue contained 403 turns). The figures for the human data correspond to the average for the five dialogues constituting the pilot portion of our

dataset. Averages are followed by the standard deviation in parentheses. The code CL means CodeLlama; L32 means Llama 3.2. The model names are followed by their size (1B, 7B, 13B). The number after P indicates the prompt number; P0 indicates that there was no prompt, the model was only given the dialogue history. The number after C indicates the length, in turns, of the dialogue history. The number after FS indicates the length, in turns, of the few-shot example. This is followed by a code indicating the particular example used ("synth" for the example created by the researchers; "real" for a real example from the dataset, followed by "b" or "e" to indicate whether the example was from the initial or a later part of the dialogue, respectively. Note that not all the prompts listed in Appendix C were analysed in this way, as the brief qualitative analysis sufficed to extract conclusions about them. For brevity, as well, not all features analysed are included here, only those relevant to the topics discussed in this paper.

B Supplementary material: Features for inferential analysis

All our inferential tests were done using logistic regression models. Thus our null hypothesis is "the predictor variable has no effect on the response variable". As per our research questions, our response variables are Coherence, Collaborativenes, and Humanness. Our main predictor variable was the source of the dialogue response: human, GPT, or Llama - for Coherence, this predictor was simplified as human vs. AI, since both models behaved similarly in terms of Coherence. We also added the response variables are predictors to see how they related to each other (we however removed Coherence as a predictor for Humanness to simplify our model when it did not converge). We extracted additional features from the responses to use as predictors, in order to better understand which factors play a role in raters' judgements. These features later also helped us understand raters' comments about their judgements on Humanness, as some of the aspects they commented on matched our features. We extracted confusion matrices linking our features to the response variables, and compared means and ranges for continuous variables. With that information, we decided which features to include in the inferential models, based on how strongly related to the predictors they seemed. The features are as follows:

- Presence of LLM-style phrases (in our case, variations of "Here's the code/solution").
- Presence of markers of spoken language (filler sounds like "uh", etc., or sentences starting with "So,"
- Response similarity with last turn. We measured sentence similarity using Google's Universal Sentence Encoder via Martino Mensio's wrapper.
- Response similarity with the human ground truth. Naturally, the value was 1 (maximum similarity) for human responses, so these were excluded from the model that tested this feature.
- Similarity of the response code with the code in the last turn. We measured similarity with the difflib Python library.
- Response length, measured in characters.
- Presence of CODE STATE label. This label was used to format the code in the responses and the dialogue history. We introduced this label in all human responses and instructed the LLMs to use it, but they did not always do that.
- Whether the rater judged correctly the source of the response (human or AI).

We also analysed where in the dialogue the response came from (e.g., middle, very end, etc.), though this did not seem to play a role. Additionally, the models included as random effects the rater and the sample ID: we included the former because of the variability observed among raters. Raw agreement percentages were 82.48% for Coherence ratings, 76.32% for Collaborativeness, and 84.00% for Humanness, but the classes are imbalanced [70% of answers were rated coherent, 43% as collaborative and 22% as NOT collaborative, and 60% as human], resulting in low agreement measures if the class distribution is taken into account. Our decision to include raters as random effects was further justified when we observed that half the times that a human response was misjudged as coming from an AI, this was due to a specific annotator being misled by the formatting of the answers. We included the latter random effect, sample ID, because we also had an imbalanced distribution: all samples, consisting of the set of three responses

Approach	Word count	LLM phrases	Human interjections
Human	26.88 (33.41)	0.0054 (33.41)	1.35 (1.78)
CL 7B P0 C4	308.08 (63.42)	3.06 (1.63)	4.27 (3.62)
CL 7B P1 C4	324.31 (54.39)	2.81 (1.53)	4.53 (3.27)
CL 13B P0 C4	312.02 (62.14)	2.51 (1.49)	5.11 (3.95)
L32 1B P1 C4	380.73 (51.24)	0.07 (0.25)	8.98 (5.74)
L32 1B P7 C4	337.77 (121.12)	0.35 (0.76)	8.71 (5.47)
L32 1B P8 C4	140.8 (95.61)	0.31 (0.68)	2.77 (3.19)
L32 1B P9 C8	114.13 (56.56)	0.16 (0.46)	2.72 (2.66)
L32 1B P9 C50	104.68 (37.1)	0.03 (0.17)	2.70 (2.48)
L32 1B P11 C4 FS27-synth	136.88 (97.64)	0.3 (0.67)	2.46 (2.56)
L32 1B P11 C4 FS50-synth	141.31 (90.87)	0.28 (0.67)	3.55 (12.7)
L32 1B P12 C4 FS50-synth	152.29 (101.23)	0.39 (0.79)	2.60 (2.49)
L32 1B P12 C50 FS27-synth	107.82 (63.29)	0.04 (0.26)	2.70 (2.50)
L32 1B P12 C50 FS27-real-b	108.86 (66.19)	0.02 (0.16)	2.87 (2.44)
L32 1B P12 C4 FS50-real-b	155.98 (109.56)	0.47 (0.77)	3.62 (4.29)
L32 1B P12 C50 FS50-real-e	143.41 (92.45)	0.43 (0.76)	2.90 (2.87)

Table 5: Average results (and standard deviation) per response for each approach (model + prompting strategy) of the preliminary analysis.

(human, GPT, Llama) for a particular query point, were rated by at least two raters, but the number of ratings ranged up to eleven. Below are the models on which we base our Results section; as readers can see, we implemented them using the glmer and clmm libraries in R, depending on the type of response variable; those libraries use Wald tests to obtain the p values:

- Coherence:

```
glmer(Coherence_B ~ realHumanness
+ Collaborativeness_O + Human_or_AI_B
+ Markers_LLM
+ Similarity_with_last_turn
+ Code_similarity_with_last_turn
+ (1|Annotator) + (1|Sample_ID),
data = human_clean, family = binomial)
```

- Collaborativeness:

```
clmm(Collaborativeness_O
~ Source_F + Coherence_B
+ Human_or_AI_B + Markers_LLM
+ Markers_spoken_language
+ Response_length_S
+ Code_similarity_with_last_turn
+ (1|Annotator) + (1|Sample_ID),
data = human_clean)
```

- Humanness:

```
glmer(Human_or_AI_B ~ Source_F
+ Collaborativeness_O
+ Markers_LLM
+ Markers_spoken_language
+ Has_CODE_STATE
+ did_annotator_guess
+ Code_similarity_with_last_turn
+ (1|Annotator),
data = human_clean, family = binomial)
```

- Effect of AI answers' similarity with ground truth on Humanness:

```
glmer(Human_or_AI_B
~ Similarity_with_GT
+ (1|Annotator) + (1|Sample_ID),
data = human_clean_onlyAI_n2,
family = binomial)
```

As we performed multiple tests, we adjusted our alpha using Bonferroni correction. The adjusted significance threshold is p < $1.8e^{-3}$.

The main result with statistical significance is the effect of the response source (human, Llama, GPT) on Humanness ratings. If we convert log odds to probabilities for easier interpretation, while human answers have a 97.82% probability of being rated as human, for Llama answers this is 72.88%, and 40% for GPT answers (p = $1.51e^{-4}$ for human ground truth, p = $5.9e^{-13}$ for GPT, p = $5.84e^{-10}$ for Llama).

C Supplementary material: Prompts

Below are the prompts that we used. The one highlighted in yellow is the one that was used for evaluation. As discussed in Section 3.2, we tested the prompts with varying context-window lengths (4, 8, 50). The initial prompts included the programming task description inside the dialogue history, as it was included in the students' code as comments. Later prompts removed this from the code and presented it only once in the prompt.

PROMPT CHARACTERISTICS:
Focus on the tone of the response and it being only one turn. Includes description of CODE STATE label.
PROMPT:
You are a pair programming partner.
Continue the dialogue with ONE utterance in a tone suitable to the dialogue history.
You are the user's peer, so you know Python, but you're not the user's teacher, you're learning together; you're equal partners, so don't be too much of a people pleaser. You want the user to be able to think and contribute equally; admit when you're not sure how to continue, instead of misleading. Finish your turn including the CODE STATE; if you want to make changes in the code, update the CODE STATE.
The content of CODE STATE should be the last part of your turn, as you can see in the dialogue history turns.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 1: Prompt 01

PROMPT CHARACTERISTICS:
Focus on the tone of the response and it being only one turn. Emphasis on the response having to be to the dialogue history. Includes description of CODE STATE label.
PROMPT:
You are a pair programming partner.
Continue the dialogue from the history with ONE utterance in a tone suitable to the dialogue history.
You are the user's peer, so you know Python, but you're not the user's teacher, you're learning together; you're equal partners, so don't be too much of a people pleaser.
You want the user to be able to think and contribute equally; admit when you're not sure how to continue, instead of misleading.
Finish your turn including the CODE STATE; if you want to make changes in the code, update the CODE STATE.
The content of CODE STATE should be the last part of your turn, as you can see in the dialogue history turns.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 2: Prompt 02

PROMPT CHARACTERISTICS:
Focus on agent persona.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 3: Prompt 03

PROMPT CHARACTERISTICS:
Focus on agent persona. Brief.
PROMPT:
You are a university student learning Python by pair programming with a peer.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 4: Prompt 04

PROMPT CHARACTERISTICS:
Combination of focus on response characteristics and agent persona. Brief.
PROMPT:
You are a university student learning Python by pair programming with a peer.
You respond to your fellow student with one turn, which may contain an utterance and maybe a change to the code, or not.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 5: Prompt 05

PROMPT CHARACTERISTICS:
Focus on agent persona. Brief. Contextualisation of context.
PROMPT:
You are a university student learning Python by pair programming with a peer.
The following is some context from your conversation, including code, which may or may not change between turns.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 6: Prompt 06

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
The following is some context from your conversation, including code, which may or may not change between turns.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 7: Prompt 07

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context. Description of CODE STATE label.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
The following is some context from your conversation, including code, which may or may not change between turns.
The code appears after the CODE STATE tag; you can see that there's no need to preamble it, you and your partner are both aware of that tag.
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 8: Prompt 08

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context. Description of CODE STATE label. Task instructions separated from dialogue context. Request json output.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
The following turns are some context from your conversation, including code, which may or may not change between turns.
The code appears after the CODE STATE tag; you can see that there's no need to preamble it, you and your partner are both aware of that tag.
If there are no changes in the code, it is the same as in the previous turn.
Output your response in json format with a 'response' key and a 'code' key (which can be empty if there's no code to return).
Before seeing the dialogue context, here is the task instructions, formatted as comments:
<TASK INSTRUCTIONS>
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 9: Prompt 09

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context. Description of CODE STATE label. Task instructions separated from dialogue context. Request json output; example included.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
The following turns are some context from your conversation, including code, which may or may not change between turns.
The code appears after the CODE STATE tag; you can see that there's no need to preamble it, you and your partner are both aware of that tag.
If there are no changes in the code, it is the same as in the previous turn.
Output your response in json format with a 'response' key and a 'code' key (which can be empty if there's no code to return).
For example, you can return {'response':'What about using a for loop?','code':'for i in our_list:'}.
Before seeing the dialogue context, here is the task instructions, formatted as comments:
<TASK INSTRUCTIONS>
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 10: Prompt 10

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context. Description of CODE STATE label. Task instructions separated from dialogue context. Request json output; example included. Few-shot example included.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
Below is an example excerpt from conversation between two students in the same setting, solving a different task:
EXAMPLE TASK: <FEW-SHOT EXAMPLE>
As in the example, respond following the context below. You see that the code appears after the CODE STATE tag;
you can see that there's no need to preamble it, you and your partner are both aware of that tag.
You can also see that, if there are no changes in the code, it is the same as in the previous turn.
As in the example response, output your response in json format with a 'response' key and a 'code' key (which can be empty if there's no code to return).
For example, you can return {'response':'What about using a for loop?','code':'for i in our_list:'}.
Before seeing the dialogue context, here is the task instructions, formatted as comments:
<TASK INSTRUCTIONS>
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 11: Prompt 11

PROMPT CHARACTERISTICS:
Focus on agent persona. Contextualisation of context. Description of CODE STATE label. Task instructions separated from dialogue context. Few-shot example included in json format.
PROMPT:
You are a university student pair programming with another university student.
You are studying some computing modules at a distance university and have been paired with another student who is studying the same or similar modules.
You and your partner, as students, want to use this pair-programming session as an opportunity to learn and collaborate with a peer.
You both know some Python, but you're still learning, and you may have used it for different things in class if you're studying different modules.
As a distance-learning student, you may not fit the usual demographics for undergraduate students, and having agreed to participate in this session shows that you're eager to practice some Python and interact with a peer.
Below is an example excerpt from conversation between two students in the same setting, solving a different task:
EXAMPLE TASK: <FEW-SHOT EXAMPLE>
As in the example, respond following the context below. You see that the code appears after the CODE STATE tag;
you can see that there's no need to preamble it, you and your partner are both aware of that tag.
You can also see that, if there are no changes in the code, it is the same as in the previous turn.
Before seeing the dialogue context, here is the task instructions, formatted as comments:
<TASK INSTRUCTIONS>
<DIALOGUE HISTORY>
<DIALOGUE TURN TO RESPOND TO>

Figure 12: Final prompt