Reference Processing in Pair-Programming Dialogue

Cecilia Domingo[1], Paul Piwek[1], Michel Wermelinger[1]; Svetlana Stoyanchev[2]

[1]The Open University, Milton Keynes, United Kingdom
[2]Toshiba Europe Ltd., Cambridge, United Kingdom
{cecilia.domingo-merino, paul.piwek, michel.wermelinger}@open.ac.uk,
svetlana.stoyanchev@toshiba.eu

Introduction

Processing and generating referring acts constitutes a basic Natural Language Processing (NLP) task. However, early work focused on monologic, monomodal text (Poesio et al., 2023). Fortunately, advances in language modelling have increased the models' ability to capture complex dependencies and process different types of inputs. Instruction-tuned Large Language Models (LLMs) have brought about a paradigm shift whereby complex tasks seem to be solvable through mere intention, expressed via natural-language prompts (Sarkar, 2024). However, LLMs reveal numerous limitations, and prompt and context engineering need to be carefully executed. LLMs nonetheless open the door for research in more complex NLP tasks and settings. Pair Programming[1] (PP) is one of such complex settings, as it involves multimodal task-oriented dialogue in a dynamic environment, as well as special terminology. Since PP has proven to be very effective in both professional and educational settings (Hawlitschek et al., 2023), there has been interest in developing AI agents to facilitate it (Robe et al., 2020; Kuttal et al., 2021). To out knowledge, no such system has yet been developed[2]. Can LLMs bring this idea to reality? To answer that, we first need to assess whether they possess the basic understanding capabilities to engage in dialogue in this setting.

Research Question

Referring acts in PP pose a special challenge, due to multimodality, a dynamic environment, and morphologically idiosyncratic domain terminology. Multimodality is inherent to PP dialogue, as it involves both verbal communication and the creation of code; speakers can thus use visual forms of reference, such as pointing at the code with the cursor or a finger. PP takes place in a dynamic environment because, throughout the dialogue, the speakers construct the code entities that they can refer to. Unlike other domains, in PP the entities that will be mentioned are not previously known – even when the programming task is known beforehand, there are countless approaches to solve it. Moreover, most entities in PP appear as variables, whose name, value, or location in the code can change. Unlike other co-construction tasks where the building blocks are tangible (e.g., Kontogiorgos et al., 2018), in PP the code elements can be discussed verbally before any code is typed into existence. Finally, PP makes use of specialised terminology. PP terminology can use any part of speech to denote objects (e.g., "a for" is no longer a preposition, but a type of iteration), and there is great flexibility in the form of proper names in this domain (e.g., a variable can be called "usr_score" or simply "x"). With these challenges in mind, we pose the following question:

[1] PP is a collaboration technique where two programmers work together on the same piece of code.
[2] Some claims have been made about using LLMs for PP, but these confuse the role of a PP partner with that of a tutor or an assistant.

How do LLMs process and generate references in pair-programming dialogue?

Impact: By exploring this question, we aim to provide insights into LLMs' linguistic capabilities in complex dialogue settings. These shall also pave the way for the development of AI agents that can engage in PP or other types of task-oriented and/or situated dialogue. With our experiments, we also aim to contribute insights on prompt engineering and context engineering to facilitate the use of LLMs for linguistic research. Moreover, given the lack of available multimodal PP data, we collected our own dataset, which we shall make available upon project completion for further linguistic and educational research.

Methodology

Our first step was collecting a dataset or PP dialogues. We recorded 25 pairs of students pair-programming remotely and collected several types of data: speech (transcribed semi-automatically), screen recordings, keylog records, and code files produced. We then had the references annotated, which involved selecting mentions, classifying them according to linguistic form, multimodality, and type of code element being mentioned, linking them in coreference chains, and locating the code being mentioned in the code files. We developed an annotation scheme following an iterative (Fuoli et al., 2018) and collaborative (Godwin & Piwek, 2016) methodology. We trained a total of seven annotators on the different annotation tasks and completed 2-5 validation rounds. Final agreement varied depending on task difficulty and subjectivity; it was lower on tasks involving linking (e.g., 47% agreement and 0.01 α on referent code span), and high on classification tasks (e.g., 71% perfect-match agreement on mention detection and linguistic classification, with 0.89 κ); but the collaborative approach allowed us to understand that most disagreements stemmed from the inherent ambiguity of referring acts. Ambiguity has been observed to be very common in referring acts in dialogue (Poesio et al., 2023); we must thus consider it in evaluation by establishing reasonable baselines and carrying out error analyses.

We are currently carrying out experiments testing models of different sizes from the Llama and GPT families, as representative and accessible examples of leading LLMs suited for code and for which there are research insights we can draw upon. We are working on mention detection, coreference resolution, and code-entity linking. Even though LLMs appear as an easy end-to-end solution to many NLP tasks, recent studies suggest that, even with this technology, a modular approach is preferable (Liu et al., 2024; Manikantan et al., 2024). We are thus establishing end-to-end reference processing as a low baseline and testing multi-step approaches for each subtask – e.g., Manikantan et al. (2024) showed that LLMs' poor performance in mention detection can be substantially improved by dividing the task into mention-head detection and mention-head expansion. Working with LLMs, two key aspects are prompt engineering and context engineering. Therefore, before evaluating the key approaches on our evaluation set, we are testing different prompting approaches and context types on the development set. For example, our previous work suggested the importance of balancing desired output details with a persona-based prompt (White et al., 2023), which harnesses the model's large knowledge base about what may be associated with that persona. Context engineering is especially important for our coreference resolution task, where we need to explore the role of multimodal input for mutual disambiguation: in cases where references are very ambiguous, data from the keyboard, mouse or screen could improve performance – e.g., see Chai et al., (2004) for an example of how multimodal data can improve performance where another input component fails.

For evaluation, we are adopting different measures that are most informative for each task: F1 scores for mention detection, the three standard CONLL metrics for coreference resolution, and accuracy for code-entity linking, looking also at partial matches. We plan to carry an error analysis to assess whether low performance metrics may be caused by ambiguous references; the analysis, powered by the detailed annotations of our dataset, can also shed light on which of the idiosyncrasies of PP dialogue pose the biggest challenges.

Preliminary Findings

We analysed a total of 22 dialogues (~30 minutes each). This allowed us to study the relevance of referring acts in PP and how the idiosyncrasies of this communicative setting are reflected on the references. We made the following initial observations:

Referring acts are a key element in PP dialogue, and they can't be disambiguated in isolation: Mentions are very common, amounting on average to around a third of a dialogue. We annotated both singletons and references that are part of coreference chains; 44% of references are in coreference chains. This shows that the dialogue history will be very important for almost half the references – after a referent is activated (using the terminology from Gundel et al. [1993]), mentions with a semantically poorer form may be used, and so the context becomes even more important. In the example below, the "string" entity is introduced on turn 1 via a type identifier ("a, a string") and is then referred to with a pronoun ("it") on turn 5.

1. A: Can we, uh, I don't know, define a, a string, maybe the, the so-cool string.
2. B: Uh... Yeah, that seems like a good place to start. And then we can kind of maybe try and split it up into the.
3. A: Yeah. Yeah. So should I start defining these, this string?
4. B: Yeah, sure. Sounds good.
5. A: Um. Uh, how should I, uh, call it? Uh... Just. Um, sentence. [B types 'text'] Oh, text. Yeah, text.

The role of multimodality in referring acts is highly variable: On average, 7.7% of mentions contain a multimodal element, be it through the keyboard or the mouse. This suggests that, while multimodal input might not be as essential as expected, it could improve model performance in several instances. The high speaker variability in this regard supports this hypothesis: in one dialogue, 21.3% of mentions featured multimodality, so this input cannot be disregarded.

Processing references in PP requires understanding domain terminology and tracking changing code entities: Almost half the mentions (46.7%) refer to code. Of those mentions, half (23.4% of the total) refer to variables, so the referent may be very dynamic and its form unusual, as discussed above. Another important observation pertaining the dynamic environment is the frequency of mentions to code that has not been implemented yet. Of all references to code, 40.1% refer to code that only comes into existence later – e.g., in the excerpt above, we see the speakers discuss a string and only type it on turn 5. This suggests that the more discourse-focused task of coreference resolution could lend important support to code-entity linking.

In general, we observe that referring acts in PP are significantly impacted by the idiosyncrasies of this domain, and that has significant implications for the kind of input that LLMs require to process them.

References

[1] Chai, J. Y., Hong, P., & Zhou, M. X. (2004). A probabilistic approach to reference resolution in multimodal user interfaces. *Proceedings of the 9th International Conference on Intelligent User Interface - IUI '04*, 70. https://doi.org/10.1145/964442.964457

[2] Fuoli, M. (2018). A stepwise method for annotating appraisal. *Functions of Language*, *25*(2), 229–258. https://doi.org/10.1075/fol.15016.fuo

[3] Godwin, K., & Piwek, P. (2016). Collecting Reliable Human Judgements on Machine-Generated Language: The Case of the QG-STEC Data. *Proceedings of the 9th International Natural Language Generation Conference*, 212–216. https://doi.org/10.18653/v1/W16-6634

[4] Gundel, J., Hedberg, N., & Zacharski, R. (1993). Cognitive Status and the Form of Referring Expressions in Discourse. *Language*, *69*(2), 247–307.

[5] Hawlitschek, A., Berndt, S., & Schulz, S. (2023). Empirical research on pair programming in higher education: A literature review. *Computer Science Education*, *33*(3), 400–428. https://doi.org/10.1080/08993408.2022.2039504

[6] Kontogiorgos, D., Sibirtseva, E., Pereira, A., Skantze, G., & Gustafson, J. (2018). Multimodal Reference Resolution In Collaborative Assembly Tasks. *Proceedings of the 4th International Workshop on Multimodal Analyses Enabling Artificial Agents in Human-Machine Interaction*, 38–42. https://doi.org/10.1145/3279972.3279976

[7] Kuttal, S. K., Ong, B., Kwasny, K., & Robe, P. (2021). Trade-offs for Substituting a Human with an Agent in a Pair Programming Context: The Good, the Bad, and the Ugly. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–20. https://doi.org/10.1145/3411764.3445659

[8] Liu, X., Liu, Y., Zhang, K., Wang, K., Liu, Q., & Chen, E. (2024). OneNet: A Fine-Tuning Free Framework for Few-Shot Entity Linking via Large Language Model Prompting. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 13634–13651. https://doi.org/10.18653/v1/2024.emnlp-main.756

[9] Manikantan, K., Toshniwal, S., Tapaswi, M., & Gandhi, V. (2024). Major Entity Identification: A Generalizable Alternative to Coreference Resolution. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 11679–11695. https://doi.org/10.18653/v1/2024.emnlp-main.652

[10] Poesio, M., Yu, J., Paun, S., Aloraini, A., Lu, P., Haber, J., & Cokal, D. (2023). Computational Models of Anaphora. *Annual Review of Linguistics*, *9*(1), 561–587. https://doi.org/10.1146/annurev-linguistics-031120-111653

[11] Robe, P., Kaur Kuttal, S., Zhang, Y., & Bellamy, R. (2020). Can Machine Learning Facilitate Remote Pair Programming? Challenges, Insights & Implications. *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–11. https://doi.org/10.1109/VL/HCC50065.2020.9127250

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–6010.

[13] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*. http://arxiv.org/abs/2302.11382