Annotation Needs for Referring Expressions in Pair-Programming Dialogue

Cecilia Domingo, Paul Piwek, Michel Wermelinger
The Open University
Milton Keynes, England
cecilia.domingo-merino@open.ac.uk
paul.piwek@open.ac.uk
michel.wermelinger@open.ac.uk

Svetlana Stoyanchev
Toshiba Europe Ltd.
Cambridge, England
svetlana.stoyanchev@toshiba.eu

Abstract

Referring expressions are a widely researched area in linguistics, with some work also on dialogue. As dialogue technology evolves and language models increasingly incorporate programming languages, pair-programming dialogues become a promising area of research. We recorded 24 dialogues between adult learners to analyse how they refer to the code that they create through the task. These dialogues present some challenges (and interesting avenues for research) for annotation and analysis, due to several factors: high multimodality, mix of abstract and specific entities in the same domain, variability in naming practices, and referents being sometimes located in the future or remaining hypothetical.

1 Introduction

Referring phenomena have long been studied in computational linguistics. While much focus was placed on monologic text, a lot of research has also been carried out in dialogue, though often in simulated settings. In our work we gather data on a real educational task solved by learners through dialogue: Python programming problems solved via pair-programming[1]. Through our data we hope to shed light on how speakers discuss code entities, in a domain whose importance is increasing in NLP as models become better at processing code along with natural language (Wan et al., 2024).

2 Remote pair-programming dialogues

We focus our research in the domain of pair programming, due to its pedagogical value (Hanks et al., 2011) and relevance in computational linguistics as language models develop increasing code capabilities (Wan et al., 2024). Even though this domain has been widely researched, most studies focus on the code produced rather than the dialogue, and thus little dialogue data is available. We recorded 24 pair-programming sessions in a remote setting (5 of them, the pilot, with simulated remoteness, i.e., participants in adjoining rooms). Participants communicated via voice call and worked together on beginner-level Python tasks using Visual Studio Code with the Live Share plugin, which allows simultaneous editing of the code file, with each user connected having their own cursor. The sessions were around 30 minutes each. The participants were 45 students (from Bachelor to PhD) and 2 staff, ages ranging from 23 to 70, and a gender split of 12 female/35 male. We recorded the dialogues, as well as the code produced at each moment and participants' mouse and keyboard activity; we also recorded the screen for additional context.

[1]Pair programming is a collaboration technique where two people work together, simultaneously, on the same piece of code.

3 Annotation needs

Several annotation schemes exist for referring expressions, most famously Ontonotes (Weischedel et al., 2013). This scheme's main appeal is its simplicity, which leads to high inter-annotator agreement, but it is also its main point of criticism, as it fails to capture phenomena that may be important (Zeldes, 2022). Other schemes (Poesio et al., 2024) tackle some of these limitations, offering useful descriptions of complex types of referring expressions. Such extensive analysis of anaphora characteristics might not be advisable in our domain, where not even a more basic one has been carried out yet. Moreover, efforts in linguistic analysis might be best spent on other aspects where our domain is more unique, such as the characteristics of the names given to code elements by coders, and how these evolve through dialogue.

3.1 Multimodality

Remote dialogues may normally be an activity that's primarily linguistic (Clark, 2005); however, the addition of the co-creation of code turns it into a highly multimodal activity. Other schemes' focus on discourse makes them unfit for our setting, where the main goal is linking references and referents across modalities. As the code becomes arguably as important as the discourse, we need to annotate both anaphora and deictic references, linking together discourse elements through coreference chains, but also linking those chains to entities in code files. In pair-programming dialogue, referents and referring acts will not only be found within the discourse; speakers will also refer to entities in the code that they are creating, and may use the mouse and keyboard to bring them into focus.

Some studies have been carried out on the use of pointing gestures for referring and show that they play an important role: e.g., gestures can replace locative expressions (Kehler, 2000), or they can contribute to mutual disambiguation, i.e., the ambiguity in discourse can be cleared with the information from other modalities and vice versa (Kaiser et al., 2003). In some settings, pointing gestures accompanied only few utterances (e.g., 16% of utterances (Sluis et al., 2008), though this number is not insignificant). However, in our preliminary analysis of publicly available data[2] 51% of utterances that mentioned code (which were 55% of the total) featured the mouse pointer or keyboard playing a role in the referring expression — as far as the video data allowed us to see. It is also important to note that different speakers show different strategies in their use of pointing (Piwek, 2007); such variability might also be observable in the use of the mouse and keyboard as pointing devices. Moreover, the literature has mostly focused on hand pointing — we might expect different behaviours regarding other forms of pointing (e.g., with the cursor).

3.2 Abstract and unrealised entities

Several studies on referring expressions have been carried out in highly controlled settings where the entities mentioned and their features are previously known to the researchers who placed them in the setting (Koolen, 2013; Rubio-Fernández, 2024). In our setting, most entities do not exist until the speakers create them into the code — they begin with a blank canvas. As the entities of the dialogue are created through it and not known beforehand, we need to annotate their characteristics post factum, distinguishing between abstract discussions of code and mentions to specific entities in the speaker's code files. See the example utterances below (from different dialogues) where we contrast a reference to an array as an abstract programming entity with an array as a specific entity present in the code:

- **Abstract array:** I can't remember how you do an array.
- **Array in the code:** I think that, that does need to be kind of an array so that I think that does need to be in square brackets.

Yet another peculiarity of the pair-programming task is that, as the speakers must discuss the problem to reach an agreement on how to solve it, often entities may be mentioned only to later be discarded as the discussion brings forth better solutions. In some cases the entity might be preserved, but the speakers may still discuss it at length before finally implementing it, thus speaking about a concrete but unrealised entity. In the excerpt below, speakers A and B discuss a string that they are going to type, and they give it a name, but the string does not become realised in the code until turn 5, when they change the name to 'text':

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.

4 Conclusions

Pair-programming dialogues possess several characteristics that set them apart from other dialogue domains studied so far, and thus require custom tools for their annotation and analysis. It is a highly multimodal setting that requires linking discourse

[2] http://www.pairwith.us/tv

to another modality (code) and taking note of accompanying gestures in a less-studied form (pointing using the mouse and keyboard). As it is a dynamic environment (Kumar et al., 2022) where entities are created through the dialogue, we need to characterise those entities post factum, observing as well whether they can be currently linked to the code, refer to future code, or remain hypothetical.

Acknowledgments

This work has financial support from EPSRC Training Grant DTP 2020-2021 Open University and Toshiba Europe Limited. This research project was reviewed by, and received a favourable opinion from, our university's Human Research Ethics Committee.

References

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

Ed Kaiser, Alex Olwal, David McGee, Hrvoje Benko, Andrea Corradini, Xiaoguang Li, Phil Cohen, and Steven Feiner. 2003. Mutual dissambiguation of 3d multimodal interaction in augmented and virtual reality. In *Proceedings of The Fifth International Conference on Multimodal Interfaces (ICMI '03). Vancouver, BC. Canada*, pages 12–19.

Andrew Kehler. 2000. Cognitive status and form of reference in multimodal human-computer interaction. In *Proceedings of the seventeenth national conference on artificial intelligence*, pages 685–690. The AAAI Press, Menlo Park, California.

Ruud Martinus Franciscus Koolen. 2013. Need I say more? On overspecification in definite referenc. Unpublished PhD Thesis, OCLC: 856996240.

Abhinav Kumar, Barbara Di Eugenio, Abari Bhattacharya, Jillian Aurisano, and Andrew Johnson. 2022. Reference resolution and context change in multimodal situated dialogue for exploring data visualizations. *Preprint*, arxiv:2209.02215 [cs].

Paul Piwek. 2007. Modality choice for generation of referring acts: Pointing versus describing. In *Proceedings of Workshop on Multimodal Output Generation (MOG 2007)*, pages 129–139.

Massimo Poesio, Maris Camilleri, Paloma Carretero Garcia, Juntao Yu, and Mark-Christoph Müller. 2024. The ARRAU 3.0 corpus. In *Proceedings of the 5th Workshop on Computational Approaches to Discourse (CODI 2024)*, pages 127–138. Association for Computational Linguistics.

Paula Rubio-Fernández. 2024. Referential efficiency as speaker-listener coordination. 2024 CORE Project Workshop.

Ielka van der Sluis, Paul Piwek, Albert Gatt, and Adrian Bangerter. 2008. Towards a balanced corpus of multimodal referring expressions in dialogue. In *Proceedings of the Symposium on Multimodal Output Generation*.

Yao Wan, Zhangqian Bi, Yang He, Jianguo Zhang, Hongyu Zhang, Yulei Sui, Guandong Xu, Hai Jin, and Philip Yu. 2024. Deep learning for code intelligence: Survey, benchmark and toolkit. *ACM Computing Surveys*, pages 1–39.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes release 5.0.

Amir Zeldes. 2022. Opinion piece: Can we fix the scope for coreference?: Problems and solutions for benchmarks beyond OntoNotes. *Dialogue & Discourse*, 13(1):41–62.