Mention detection with LLMs in pair-programming dialogue

Cecilia Domingo, Paul Piwek, Michel Wermelinger
The Open University
Milton Keynes, England
cecilia.domingo-merino, paul.piwek, and michel.wermelinger (@open.ac.uk)
and **Svetlana Stoyanchev**
Toshiba Europe
Cambridge, England
svetlana.stoyanchev@toshiba.eu

Abstract

We tackle the task of mention detection for pair-programming dialogue, a setting which adds several challenges to the task due to the characteristics of natural dialogue, the dynamic environment of the dialogue task, and the domain-specific vocabulary and structures. We compare recent variants of the Llama and GPT families and explore different prompt and context engineering approaches. While aspects like hesitations and references to read-out code and variable names made the task challenging, GPT 4.1 approximated human performance when we provided few-shot examples similar to the inference text and corrected formatting errors.

1 Introduction

Pair programming is a collaboration technique which has received a lot of scholarly attention due to the numerous benefits it can lead to, such as improved confidence and code quality (Hawlitschek et al., 2023). It involves two programmers working together on the same piece of code. The setting may vary (e.g., co-located or distributed pair programming); the pair dynamics may also vary: e.g., scholars mostly observe a navigator and a driver role, but these may switch variably during the session, and some scholars also observe different roles (Hanks et al., 2011). However, one aspect remains constant: dialogue drives the task. Dialogue complicates NLP tasks by introducing new challenges not found in the more traditionally studied written genres, and the idiosyncrasies of pair-programming dialogue further add to those challenges.

Below we present a short excerpt from our dataset to illustrate the type of dialogue that we are working with. In this excerpt, we can see some general characteristics of dialogue, such as hesitations (e.g., the repetition of determiners on the first line or the numerous filler sounds on the last line) or incomplete sentences (e.g., the turn in line 2 ends abruptly). We also observe some characteristics more unique to our type of setting, such as the use of domain terminology (e.g., here 'a string' is not thin rope) and references to unrealised entities (e.g., the speakers keep mentioning a string but they only type it into the code with the name 'text' on the fifth turn. This is frequent in this type of dialogue because the collaborative setting makes it necessary to discuss ideas with one's partner before deciding what to put into practice.).

A: Can we, uh, I don't know, define a, a string, maybe the, the so-cool string.
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.
A: Yeah. Yeah. So should I start defining these, this string?
B: Yeah, sure. Sounds good.
A: Um. Uh, how should I, uh, call it? Uh... Just. Um, sentence. [B types 'text'; the name 'sentence' is discarded and entity becomes realised as 'text'] Oh, text. Yeah, text

In this work we focus on mention detection, the basic pillar of work on reference (e.g., this importance has been described in terms such as 'The performance of mention detection is to this day one of the most important factors in anaphora resolution' (Poesio et al., 2023, p. 571)). In simple terms, it consists on extracting all text spans that refer to some entity in the world, be it physical or abstract, or a broader element of the discourse in the case of discourse deixis. We use 'entity' to mean anything that exists, whether concrete or abstract; thus, mentions will always refer to an entity, and sometimes they may also be linked to other mentions if they all refer to the same entity. A mention that refers to an entity that is only referred to once in the discourse is called a singleton mention. Although the basic definition of the mention detection task is rather simple, researchers often differ in the

Proceedings of the Eighth Workshop on Computational Models of Reference, Anaphora and Coreference (CRAC 2025), pages 42–54

specification of the concrete types of mentions (and references that these make) that they consider (e.g., see Zeldes (2022) for a critique of the commonly used OntoNotes schema, which includes its omission of singletons, predication, generic mentions, and nested mentions). While a full discussion of our annotation scheme and process is beyond the scope of the current paper, we do provide some key details in Section 3.1. In relation to the topic of what is considered a mention, we shall note that we included singletons (single mentions to an entity not mentioned any other time), predication (mentions equated to each other through a copula verb, like in 'Sahil is a lecturer'), generic mentions (e.g., 'Good children eat their vegetables and then Santa brings presents to good children'), and nested mentions (e.g., 'the book on the table' would be labelled as the book on the table and the table). We did however not include bridging anaphora (Clark, 1975) in our annotations[1], as we considered it much different from other types of anaphora — e.g., it is distinct enough to warrant a separate task in the CODI-CRAC task (Khosla et al., 2021; Li et al., 2021). Nonetheless, at the mention detection stage, the anaphors in bridging anaphora are still considered mentions; we simply consider them first mentions at the coreference resolution stage, instead of linking them to an existing antecedent through bridging anaphora. As we are working with LLMs, we shall rely of their vast training data to supply the schemata needed to interpret bridging anaphora — we however encourage further work in this area upon the release of our dataset[2]. With regard to discourse deixis, this is also considered a sufficiently distinct type of anaphora resolution to warrant its own task (Khosla et al., 2021; Li et al., 2021). As such, we did not include any discourse antecedents in our annotation of mentions, though for our later annotation of coreference we did add them separately after noting that discourse deixis was too frequent not to be included in the interpretation of references in our data.

[1]In bridging anaphora, the anaphor (i.e., the mention) is linked to a referent that it is not equivalent to, but from which it is inferred through shared common ground. For example, in 'I went to a Spanish restaurant. The waiter was from Cuenca', *the waiter* would be linked to *a Spanish restaurant* via bridging — it is the waiter's first appearance in the discourse, but we could already infer his existence from our knowledge of restaurants.

[2]Due to ongoing work, we are currently unable to release the dataset, but have scheduled its release for the beginning of 2026. Data will only made available upon request to avoid data contamination.

In order to analyse the characteristics of pair-programming dialogue and observe how they may impact NLP tasks related to reference, we collected and annotated a dataset of pair-programming sessions. We describe the collection and annotation procedure in Section 3.1. An analysis of our data confirmed the relevance of references and shed light on their characteristics in this domain. We then used this data to experiment with state-of-the art LLMs and measure their performance on this task, paying special attention to how the observed characteristics of pair-programming dialogue impact it. We describe our experimental methodology in Section 3.2, and then present and discuss our results in Sections 4 and 5. Our work is motivated by the ulterior goal of facilitating the development of AI agents that can act as pair-programming partners in an educational setting when no suitable human partner is available for the student to benefit from this practice, as suggested in the work of Robe and Kuttal (Kuttal et al., 2020; Robe et al., 2020; Kuttal et al., 2021; Robe, 2021; Robe and Kuttal, 2022). This influenced both the design of our data collection and experiment design: we use LLMs as the most accessible tools for dialogue agent design under the new LLM-based paradigm (Jurafsky and Martin, 2025). Our results have important repercussions for research not only on mention detection, but also on other tasks related to referring acts, as they build upon mention detection; we discuss this impact in Section 5, but it is first contextualised through the body of research we present now in Section 2.

2 Related work

Although a lot of research on reference has focused on written genres, an increasing body of research has been developed in dialogue as well, with more available datasets (Khosla et al., 2021; Li et al., 2021; Poesio et al., 2023). With the recent paradigm shift introduced by the popularisation of LLMs, research in this area has also been facilitated. With their vast training datasets and their optimisation for dialogue, these tools offer great promise for NLP tasks related to reference, even in dialogue settings. Nonetheless, initial research on coreference resolution using these types of models shows that they do not always surpass previous approaches (Mitkov and An Ha, 2024), but they offer great generalisability in unsupervised settings (Le and Ritter, 2023). The models' vast inherent knowl-

edge and their capacity for in-context learning have also been successfully harnessed for entity linking (Liu et al., 2024). One area in which knowledge is still much lacking, though, is mention detection (e.g., the works mentioned above rely on ground-truth mentions for the successful results).

Mention detection, however, cannot be taken for granted, as it is the basic task upon which all other reference tasks are built (Li et al., 2021). This task has been shown to be challenging and, therefore, attempts have been made at simplifying it. Manikantan *et al*. (2024) proposed a task that focuses on the major entities. i.e., the most frequent ones — that task is useful in their literary setting, where the main characters of a story are known, but such major entities are not so easily extracted in an online setting with a dynamic environment. Even with that simplified task, the approach also had to be broken down into steps for the models to achieve satisfactory performance; in this case, the grammatical heads of the mentions were first extracted before they were expanded into the full span of each mention. However, the full mention detection task (i.e., working on all mentions, not only those referring to the most frequent entities) still poses a big challenge. This is specially true in the domain of situated dialogue with a dynamic environment, where the system has no prior information about the entities that may be mentioned. Some evidence of these challenges are already observed in the work of Madge *et al*. (2025), who tested coreference resolution in one such environments and corroborated that performance was significantly lower than in other simpler settings — their experiments included the extracted mentions as part of the input, thus not reflecting mention detection performance, but we can expect the challenges of the dialogue setting to similarly affect mention detection.

3 Methodology

3.1 Data collection and annotation

We collected a dataset of 22 distributed (remote) pair-programming dialogues between students at our institution. We recorded a total of 25 dialogues, though two were discarded for technical reasons and a participant's withdrawal; a further dialogue is excluded, as it was used only for training annotators. Each session lasted around 30 minutes, and communication took place only via voice call and a shared programming interface. We recorded several data sources: dialogue (audio and transcription[3]), keylog records[4], video and screenshots of the programming interface, and files registering all changes to the code. The keylog records were incorporated into the json files containing the dialogue transcripts through their timestamps; however, that level of context was not used in this task, as the human annotators did not use it either for mention labelling. The keylog records and the separate visual information are employed instead for other tasks in our project for which a richer context is needed. Further details about the data recording can be found in (Domingo et al., 2024).

The dataset was then annotated by a team of 7 people trained specifically for the task; before training, they had to demonstrate the necessary linguistic and programming skills through a test or relevant qualifications. The majority of the team worked on annotating coreference chains and linking code references to code files. The task of locating mentions was carried out by the two team members with expert knowledge of Linguistics using LabelStudio.[5] The interface was configured so that no unit smaller than a word could be captured to avoid human errors, and any adjoining punctuation marks (e.g., a comma at the end of a mention) was removed during post-processing. The annotation scheme and guidelines[6] were developed through discussion among the research team validated through three rounds of the two experts double coding sections of two dialogues and discussing the process as a team with the main researcher, who performed both a quantitative and qualitative analysis of the output. We thus combined a traditional iterative development approach (Fuoli, 2018) with a socialisation-based approach (Godwin and Piwek, 2016) for improved efficiency and reliability. The annotators who labelled the mentions also classified them into the linguistic categories outlined in Appendix B. As we have discussed, a full discussion of the annotation scheme is beyond the scope of this article, though more details can be found in the supplementary materials. One important piece of information is that we in-

[3]Dialogues were transcribed using Whisper (Radford et al., 2022) and revised manually. Manual revision was necessary due to the tool's inability at the time to successfully handle disfluencies and overlapping speech, the imprecisions in audio segmentation, and the challenging domain terminology.

[4]Keylog records were obtained using a custom tool or RUI, depending on compatibility with participants' computer.

[5]https://huggingface.co/LabelStudio

[6]The section of the guidelines concerning mention detection is available as additional materials and a summary can be found on Appendix B.

cluded singletons — we want the data to be usable as training/testing for an online system, where it is not possible to know if a singleton is a singleton or part of a coreference chain until the dialogue is over.

Table 1 shows the key details of our dataset. An analysis of our data shows that around a third of all the words in the corpus correspond to mentions: the average number of mentions in our dialogues is 692, with an average number of 3445 words per dialogue and an average length of 1.6 words per mention.

Number of:	Average (per dialogue)	Total (22 dialogues)
turns	385 (SD = 72)	8468
words	3445 (SD = 768)	75797
mentions	692 (SD = 189)	15222
mentions in chains	289 (SD = 59)	6365

Table 1: Dataset details

For our experiments, we used one dialogue as our development set, and the remaining 21 for evaluation. The development set was chosen semi-randomly to ensure usefulness: we selected it randomly from the top-ten most 'average' dialogues in terms of the percentage of multimodal mentions, mentions to abstract programming concepts, names, read-out sections of code, and 'intensional' objects (borrowing the terminology from (Madge et al., 2025) to denote references to planned task outputs that have not been produced yet, or ever).

3.2 Prompt and context engineering and output processing

In recent years, numerous LLMs have been released, including many trained on programming languages in addition to natural languages (Jiang et al., 2024). It is therefore a futile attempt to try to carry out a comprehensive performance test of all possible LLMs, nor even of the most recent ones, given the rapid developments in the field. Instead, we chose representative examples to illustrate how the challenging aspects of our domain may be tackled with an LLM approach. We thus limited our experiments to recent variants of the Llama and GPT families. With our choice of models we strove to select frequently used ones — e.g., these are the families used too by Le & Ritter (2023), and they represent both proprietary and open-weights models. Our model selection was further motivated by the availability of API services that offer sufficient data protection safeguards. With regard to the model parameters, throughout our experiments, we have used a constant temperature of 0, for more deterministic, replicable results.

Prompting makes running the models easy in principle (Sarkar, 2024); however, results are highly dependent on the type of prompt used (White et al., 2023). Bearing this in mind, we tested different prompting approaches. Our initial prompt refinement was based on a qualitative analysis of 20 random outputs from each prompting approach, considering task completion, format adherence, and task accuracy. We are aware of possible hallucinations, especially with regard to numerical values, so we quickly discarded any approach reliant on index numbers or any kind of numerical identifier. Instead, we obtained more consistent results with simple XML tags (<M></M>). Previous work with LLMs (Domingo et al., in press) showed us the effectiveness of a persona-based prompt: instead of providing many details about the task we expect the model to complete, we describe a persona for it to adopt and rely on its vast training data to supply the definition of what that persona entails. A non-human persona showed the best results — the prompts can be found on Appendix A.1. In our pursuit of consistency, we did not perform many experiments with temperature parameters, selecting a temperature of 0 most of the time for consistent, replicable results. Based on the work by Manikantan et al. (2024), we also tested splitting the task into the two subtasks that they identify: mention heads are detected first, and then the second task consists one expanding them, which can be done with the same model or using SpaCy[7].

In addition to the prompt, the context also requires 'engineering'. Recent models are capable of processing long inputs, and it is sometimes the case that exploiting this capacity by adding long contexts improves performance. However, long contexts can also introduce noise and draw the model's attention away from the main instructions. Therefore, our context engineering efforts concerned not only context length, but also quality. We experimented with different few-shot settings where we provided a varying number of example dialogue turns with ground-truth labels (from 1 to 10). The

[7] https://spacy.io/

examples were randomly selected, or fabricated by us aiming to exemplify the main difficulties of our setting, or manually selected to be the most representative overall, or mixed. We also tested extracting pool of turns[8] from which the best one was retrieved for each inference turn based on sentence similarity (also using SpaCy[9]).

We paired our quantitative evaluation metrics (F1) with continuous small qualitative analyses to better understand the performance of each approach. We observed some consistent errors that could be corrected through simple rules (e.g., when the models added spaces or prefaced the output with an arrow), so we added an automated post-processing step to our pipeline. Of special interest were some inconsistencies in the models' processing of contracted verbs: e.g., we'll was sometimes returned as <M>we'll</M> and sometimes as <M>we</M>'ll; we corrected the output to follow the latter format, in line with our ground truth. The use of dialogue data added another difficulty: sometimes the models struggled with mentions broken by disfluencies. We corrected the cases where a determiner was repeated, sometimes with a filler sound in between (e.g., 'the, uh, the string'), ensuring that the mention labels grouped the two determiners in the same mention.

We carried out our initial experiments with smaller models for cost/time efficiency: Llama 3 8B and GPT 4o mini. Both models' release date is only a few months apart, and both are claimed to have a similar size (Abacha et al., 2025), though the GPT model is distilled from a larger one. Nonetheless, these two options offer the highest comparability among the ones available to us. After analysing our quantitative results, we tested the best approaches on bigger versions of the models: Llama 3 70B and GPT4o, as well as GPT 4.1. We then tested the generalisability of the approaches on the evaluation set. As our human performance ceiling we use the agreement between our annotators during the validation stages of the annotation scheme development: 82.21% to 90.39%.

[8]For our preliminary tests, we extracted a random pool as a training set. For our final experiments with the evaluation set, we were instead able to use a whole dialogue (the development data) as a more complete training set that we could expect to contain a variety of turns that could always allow to find sufficiently similar examples to the turn used for inference.

[9]Martino Mensio's Github

4 Results

Here we present the results over the evaluation data: i.e., the 21 dialogues that are not dialogue 032x028, which was used as development data The naming structure \d\d\dx\d\d\d reflects the code assigned to each speaker in the pair during anonymisation. The numbers represent the order in which people interested in participating signed the consent forms. The experiments with this data allow us to have a clearer view of model performance without a single dialogue biasing results. The design of our experiments was based on our preliminary work with the development data (dialogue 032x028)[10]. Based on preliminary work with the development data, we concluded that the most successful approach was providing few-shot examples that were similar to the turn being parsed. We used the development dialogue 032x028 as the pool from which to retrieve the few-shot examples using sentence similarity; using a whole dialogue would allow us to have a rich pool of possible examples. We also tested a zero-shot approach to have a clear view of how the few-shot examples contribute to the task. For our final experiments, we used the whole range of models available to us, both big and small: GPT 4o, GPT 4o mini, GPT 4.1, GPT 4.1 nano, Llama 3 70B, and Llama 3 8B. Under the few-shot condition, we used three few-shot examples, which had proven to be sufficient with the GPT models. However, as we had also observed that the Llama models were more sensitive to the amount of few-shot examples, we also used six few-shot examples with Llama 3 8B — given the amount of data we were testing on, we did not test with a larger number of examples, and we only used the six examples on the smaller Llama model. Also drawing on the insights from our preliminary experiments, we expected the Llama models to perform below the GPT models under any condition. Thus, with the evaluation tests we did not attempt to boost their performance closer to the GPT models; we only wished to determine to which extent increasing the number of few-shot examples boosts performance in these more context-sensitive models

Table 2 shows the GPT models' performance under the approaches we've described; Table 3 shows the performance for the Llama models. As we tested two few-shot conditions, we did not test a zero-shot approach — our preliminary experiments

[10]More details about our preliminary work can be found in Appendix A

Model	Zero-shot average			Few-shot average		
	F1pp	Ppp	Rpp	F1pp	Ppp	Rpp
GPT 4.1	0.64	0.80	0.53	0.80	0.84	0.76
GPT 4.1 nano	0.25	0.48	0.17	0.57	0.60	0.54
GPT 4o	0.27	0.53	0.18	0.70	0.72	0.68
GPT 4o mini	0.22	053	0.14	0.73	0.78	0.69

Table 2: Average performance of the GPT models across the 21 evaluation dialogues. Few-shot performance involves examples selected based on sentence similarity; in this case we use three examples. The pp suffix means that the score was obtained after post-processing the output.

made it evident that the Llama models rely to a greater extent on the few-shot examples; thus, our focus was on seeing the effect of increasing the number of examples instead.

From these tables we can observe that, as was the case with the development dialogue, the GPT models perform better across all dialogues. The best-performing model is GPT 4.1., which is meant to be suitable for very complex text tasks. Surprisingly, GPT4o mini's performance is not much lower, though it relies heavily on the post-processing of the output. As expected, the Llama models perform below the GPT models in general. We can also see that increasing the number of few-shot examples does improve performance to some extent. Here we have presented the F1 scores; however, we have also made some observations about precision and recall. For instance, the average zero-shot precision for GPT 4.1 was 0.80, but recall was 0.53; with the few-shot approach, precision increased noticeably to 0.84, but recall increased even more remarkably to 0.76. We see this tendency in the other models through both the final and preliminary experiments, where precision is higher than recall, and the difference is larger in the worse-performing approaches.

Additionally, we re-evaluated the final results using a more lenient metric, based on the work by Moosavi et al. (2019). They point out that mention detection can be evaluated based on minimum span match, instead of requiring a system to define the mention boundaries in exactly the same way as the ground truth data. They develop an algorithm for automatically extracting minimum spans from mentions without the need of additional manual annotations. Both their algorithm and a simpler method based on head extraction correlate highly with human annotations in the few datasets that include such minimum span annotations. Here we use the head extraction method for simpler implementation. Table 4 shows the adjusted average scores.

Model	FS	Few-shot average		
		F1pp	Ppp	Rpp
Llama 3 70B	3	0.59	0.62	0.56
Llama 3 8B	6	0.49	0.50	0.48
Llama 3 8B	3	0.46	0.48	0.44

Table 3: Average performance of the GPT models across the 21 evaluation dialogues. Few-shot performance involves examples selected based on sentence similarity. FS stands for the number of few-shot examples. The pp suffix means that the score was obtained after post-processing the output.

Model	Precision-pp	Recall-pp	F1-pp
GPT 4.1	0.89	0.80	0.84
GPT 4.1 nano	0.67	0.61	0.64
GPT 4o	0.76	0.72	0.74
GPT 4o mini	0.84	0.74	0.78
Llama 3 70B	0.70	0.62	0.66
Llama 3 8B	0.59	0.55	0.57

Table 4: Scores measuring minimum-span matches on the similarity-based few-shot example approach with 3 examples. The pp suffix indicates that we evaluated the output after post-processing it.

4.1 Error analysis

As our preliminary experiments covered only one dialogue, with the evaluation data we were interested in seeing how the characteristics of each dialogue affected task performance — all dialogues in this setting have some broader characteristics in common, but we already saw in previous analyses that these are displayed to different extents in each dialogue. Looking at performance across different models, we concluded that the 'easiest' dialogue was 040x054 (with a maximum F1 of 0.84 and a minimum of 0.52), and the 'hardest' was 062x059 (with a maximum F1 of 0.67 and a minimum of 0.45). Table 5 shows the main distinctive characteristics of mentions in this setting for each of these dialogues. We see that these two dialogues are in distant sides of the spectrum with regard to the percentage of mentions that are proper nouns and the mentions that are read-out code — we must also bear in mind that, in this domain, both cate-

gories are linked, as proper nouns are often variable names. To better understand performance differences, we performed a brief error analysis of these two dialogues by looking at their false negatives and their false positives; we analysed 20 mentions for each error type for each dialogue. With regard to the false negatives, in the easy dialogue, 8 of the missed mentions were direct references to code (e.g., 'for x in grade'), while this figure was 14 for the hard dialogue. In this small sample, we did not find any names in the easy dialogue's false negatives, but we found two for the hard dialogue: 'student student, test scores' and (BLEEP). These examples illustrate how names (and mentions in general) can be challenging in this domain. The first name includes a hesitation, and the second one is anonymised — though the pool of few-shot examples featured this as well. Looking at the false positives, we can observe how this task is also challenging and ambiguous for humans, as some of the false positives could be considered valid positives or even an error in the ground truth (e.g., in 'Right. I'm doing this wrong, aren't i? That's it.' the ground truth missed 'this'). In some other cases, however, the false positives cannot be considered mentions even if we apply the annotation scheme very flexibly (e.g., in the previous example, 'That's it' was returned as a mention, thus a truly false positive). We observe such verb phrases or verbs treated as mentions even when the turn shows no discourse deixis that would justify labelling a whole verb phrase as a referent; we find four cases in the easy dialogue, and two in the hard one. The main cause of false positives that we observe in both dialogues is a mismatch between the ground truth span and the output span, where the model misses parts of a mention when it is a complex noun phrase — e.g., one turn says 'Um and then we need it to output the percentage of students who passed and then output the grade', and the ground truth extracts 'the percentage of students who passed', whereas the model only extracts the main part of the phrase, 'the percentage of students'. We find four such cases in the easy dialogue, and six in the hard one. Lastly, one important source of false positives related to this is the presence of hesitations, which are common in our dataset as part of the nature of spoken dialogue; in such cases, the model can miss part of the mention or interpret a repetition as two separate mentions (e.g., in 'Yeah um maybe uh student student, test scores', the ground truth extracts 'student student, test scores' as a mention to the variable storing the student test scores, but the model extracts three separate mentions: 'student', 'student', and 'test scores').

Type of mention (percentage range)	Value for 040X054 (easy)	Value for 062x059 (hard)
Multimodality (2.82-21.36)	5.87	8.24
Abstract Mentions (0-19.67))	0.78	1.90
Names (3.84-24.29)	**8.92**	**15.96**
Read-dictate (1.39-15.35)	**6.22**	**10.47**
Intensional mentions (1.27-31.85)	5.60	3.95
Total mentions (350-1154)	482	783

Table 5: Main characteristics of the 'easiest' (040x054) and 'hardest' (062x059) dialogues. We show the types of mentions as a percentage of the total. In parentheses we show the value range across all dialogues to contextualise this data.

5 Discussion

As we mentioned in Section 3.2, agreement between our annotators for this task ranged between 82.21% and 90.39% for the three double-coding validation rounds that we ran. This shows that, even though this was one of the 'easier' tasks in our work on reference, even trained human experts sometimes disagreed. We observed some human errors in the ground truth in our error analysis (Section 4.1), but our validation work for the annotation scheme involved qualitative analyses and discussions with the annotators to ensure that there were no misalignments in their internalisation of the scheme; therefore, most disagreements at the final validation stage can be primarily attributed to inherent ambiguities of the task — while this task is simpler than the subsequent task of coreference resolution, for this latter task Poesio *et al.* (2023) noted that there is a great degree of ambiguity in references, with figures of up to 40% in dialogues annotated for discourse deixis. Bearing this in mind, we cannot expect any model to surpass the human scores; that would indicate both overfitting

to one annotator's perspective and an imbalance in our data split — we are preventing this by testing on all our dialogues, thus balancing data from both annotators. To be able to compare the models' performance with the human annotators, we calculated the models' agreement with the ground truth. For GPT4.1, this ranges between 71.86% and 88.14%; GPT 4o and GPT4o mini reach maximums of 79.86% and 81.99% respectively, but their minimums are much lower (38.08% and 37.22%). We can therefore conclude that, under the best approach, some recent models approximated human performance, but only GPT 4.1 did so consistently.

Due to the recent advances in language models, few studies exist to this date on mention detection using LLMs. However, Manikantan *et al.* (2024) and Le & Ritter (2023) offer some comparable results. The latter observed unsatisfactory performance with Instruct GPT, which yielded an F1 score of 46.5. The former, however, obtained F1 scores ranging between 77.1 and 85.5 using GPT 4 with their best approach (having the LLM extract the nucleus of the main mentions, and using SpaCy to expand the mention span). Our results are unsurprisingly higher than Le & Ritter's, possibly due to primarily their use of a less powerful model — in fact their results are similar to what we obtained with our poorest model, Llama 3 8B. Manikantan *et al.*'s better results are more similar to our best results, despite them performing a simplified task on literary texts that lack the challenges of pair-programming dialogue, probably due to our use of the latest, most powerful models. We find evidence of this in the fact that performance only reached this high range of F1 scores between 0.75 and 0.86 with GPT 4.1.

In addition to overall model performance, we have made some other key observations. We have observed through models' zero-shot performance that their base knowledge allows them to detect mentions with precision, but that in-context learning is needed for them to detect a broader range of mentions. Additionally, as we expected from analysing the characteristics of our dataset, one key issue that made mention detection difficult in this domain is the mentions to code, especially variables with their flexible form unlike that of names in other domains. Additionally, the fact that we are dealing with spoken dialogue resulted in hesitations, which also pose a significant challenge.

6 Conclusions

Through this work, we set out to explore the challenges that a pair-programming dialogue setting presents for work on reference, starting with the base task of mention detection — there can be no good coreference resolution without very good mention detection. We used LLMs for this exploration as the most recent and accessible tools for this task, imagining the kinds of tools that might available for an online pair-programming agent.

We have looked at the different experimental settings that may improve model performance in a few-shot setting: prompt and context engineering, carefully crafting suitable prompts and selecting the optimal type and amount of few-shot examples. We have observed that GPT 4.1 is close to human performance, so it could potentially replace human annotators with adequate prompt and context-engineering on texts that are not exceedingly complicated. As we have discussed throughout this work, LLMs are powerful tools that can detect mentions to some extent. However, just as humans require annotation schemes and there is often debate about which types of mentions should be included (Zeldes, 2022), the models require few-shot examples to capture the whole range of mentions required. Pair-programming dialogue presents some additional challenges that are not found in other text types, primarily hesitations stemming from it being spoken dialogue, and references to code, which involve terminology much unlike that of other domains.

Mention detection is the basic task upon which work in reference resolution is built. Therefore, insights into it are important, as outside of research scenarios coreference resolution cannot rely on having gold-standard mentions as input. We have seen some limitations from LLMs performing this task, but we have also discussed where some of these come from and how performance can be improved in many cases. These insights will inform our future work in reference resolution, and we hope it can also prove useful to the broader NLP community. As we discussed in 3.1, we recorded several types of data, and our annotation work went beyond mention detection into reference resolution and phrase grounding. We will thus work on those tasks with the outputs and insights obtained at this first stage of mention detection.

Limitations

As we have discussed in Section 3.2, we did not strive to do a comprehensive analysis of LLMs' performance that included all possible kinds of such models, nor is that feasible with the rapid developments in this area. We also did not perform a comparative analysis with other neural or rule-based approaches: our focus is on LLMs as an example of the SotA in NLP for numerous tasks, to examine to what degree the idiosyncrasies of our dialogue setting pose significant challenges. Another significant limitation of our work is the conversion of spoken dialogue into text. We used transcriptions generated with Whisper (Radford et al., 2022) and revised by us, thus having very robust verbatim transcripts. Future work with multimodal LLMs will need to determine whether the same performance is achieved when the models process the audio directly — this could hinder performance through transcription errors, or it could even simplify the task through the removal of hesitations in the initial steps of speech processing. Additionally, although our dataset is not exceedingly small, as shown on Table 1, it is very far from the size of popular datasets such as OntoNotes (Weischedel et al., 2013), which must be taken into account when interpreting our results/

Ethical considerations

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. 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 work.

References

Asma Ben Abacha, Wen-wai Yim, Yujuan Fu, Zhaoyi Sun, Meliha Yetisgen, Fei Xia, and Thomas Lin. 2025. MEDEC: A Benchmark for Medical Error Detection and Correction in Clinical Notes. *arXiv preprint*. ArXiv:2412.19260 [cs].

Herbert H. Clark. 1975. Bridging. In *Theoretical Issues in Natural Language Processing*.

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.

Cecilia Domingo, Paul Piwek, Michel Wermelinger, Svetlana Stoyanchev, Rama Doddipatla, and Kaustubh Adhikari. Human ratings of LLM response generation in pair-programming dialogue. In *Proceedings of the 18th International Natural Language Generation Conference*, Hanoi, Vietnam. Status: in press.

Matteo Fuoli. 2018. A stepwise method for annotating appraisal. *Functions of Language*, 25(2):229–258.

Keith Godwin and Paul Piwek. 2016. Collecting Reliable Human Judgements on Machine-Generated Language: The Case of the QG-STEC Data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 212–216, Edinburgh, UK. Association for Computational Linguistics.

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.

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].

Daniel Jurafsky and James Martin. 2025. Chatbots & Dialogue Systems. In *Speech and Language Processing*, pages 1–39. Stanford University.

Sopan Khosla, Juntao Yu, Ramesh Manuvinakurike, Vincent Ng, Massimo Poesio, Michael Strube, and Carolyn Rosé. 2021. The CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis in Dialogue. In *Proceedings of the CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis in Dialogue*, pages 1–15, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sandeep Kaur Kuttal, Jarow Myers, Sam Gurka, David Magar, David Piorkowski, and Rachel Bellamy. 2020. Towards Designing Conversational Agents for Pair Programming: Accounting for Creativity Strategies and Conversational Styles. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–11, Dunedin, New Zealand. IEEE.

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.

Nghia T. Le and Alan Ritter. 2023. Are Large Language Models Robust Coreference Resolvers? *arXiv preprint*. ArXiv:2305.14489 [cs].

Shengjie Li, Hideo Kobayashi, and Vincent Ng. 2021. The CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis Resolution in Dialogue: A Cross-Team Analysis. In *Proceedings of the CODI-CRAC 2021 Shared Task on Anaphora, Bridging, and Discourse Deixis in Dialogue*, pages 71–95, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Chris Madge, Maris Camilleri, Paloma Carretero Garcia, Mladen Karan, Juexi Shao, Prashant Jayannavar, Julian Hough, Benjamin Roth, and Massimo Poesio. 2025. MDC-R: The Minecraft Dialogue Corpus with Reference. *arXiv preprint*. Version Number: 1.

Kawshik Manikantan, Shubham Toshniwal, Makarand Tapaswi, and Vineet Gandhi. 2024. Major Entity Identification: A Generalizable Alternative to Coreference Resolution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11679–11695, Miami, Florida, USA. Association for Computational Linguistics.

Ruslan Mitkov and Le An Ha. 2024. Are rule-based approaches a thing of the past? The case of anaphora resolution. *Procesamiento del Lenguaje Natural*, 73(0):15–27.

Nafise Sadat Moosavi, Leo Born, Massimo Poesio, and Michael Strube. 2019. Using Automatically Extracted Minimum Spans to Disentangle Coreference Evaluation from Boundary Detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4168–4178, Florence, Italy. Association for Computational Linguistics.

Massimo Poesio, Juntao Yu, Silviu Paun, Abdulrahman Aloraini, Pengcheng Lu, Janosch Haber, and Derya Cokal. 2023. Computational Models of Anaphora. *Annual Review of Linguistics*, 9(1):561–587.

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].

Peter Robe. 2021. Designing a Pair Programming Conversational Agent. Master's thesis, University of Tulsa, Tulsa, Oklahoma.

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.

Advait Sarkar. 2024. Intention Is All You Need. In *Proceedings of the 35th Annual Conference of the Psychology of Programming Interest Group*, Liverpool.

Ralph Weischedel, Martha Palmer, Marcus Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, and Houston, Ann. 2013. OntoNotes Release 5.0. Artwork Size: 2806280 KB Pages: 2806280 KB.

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].

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.

A Appendix: Prompt and context engineering

Table 6 shows the compared performance of different models tested under the same simple setting (3 randomly selected few-shot examples; we evaluated over five folds with repeated random subsampling validation to account for variations resulting from the different random examples.). Given the size of the development set, we cannot read

Model	P	R	F1	F1pp
GPT 4.1	0.80	0.75	0.77	**0.79**
GPT 4.1 nano	0.55	0.57	0.56	**0.60**
GPT 4o	0.62	0.61	0.62	**0.66**
GPT 4o mini	0.73	0.70	0.72	**0.80**
Llama 3 8B	0.00	0.00	0.00	**0.44**

Table 6: Model comparison using 3 random few-shot examples. Initial results on the development set for P(recision,), R(ecall) and F1 after post-processing.

much into the results, but they allow us to exemplify three basic observations from our broader set of preliminary experiments: the GPT models perform markedly better than Llama, smaller distilled models are competitive with their full version, and postprocessing significantly improves results. This last point is especially true for Llama, which was consistently inconsistent in its formatting of the output.

Our initial qualitative analysis allowed us to refine our base prompt, as we discussed in Section 3.2. However, considering that the models that we are using are optimised for dialogue, we also tested a prompt that focuses more on this dialogue setting instead of a parser persona — we called this prompt a 'chatty' prompt. The different prompts are shown below. Table 7 shows the different F1 scores using our base prompt and two 'chatty' prompt versions, comparing two small models and different numbers of randomly-selected few-shot examples. The first 'chatty' prompt also follows the persona approach, encouraging the model to respond like a dialogue partner to a student; the second one introduces the setting as a dialogue, but without asking the model to adopt any human-like behaviour. As we can see, performance with the first of the 'chatty' prompts decreases noticeably, but with the second one it is similar to the base prompt. The table also allows us to exemplify the effect of the number of few-shot examples. We see that, with the GPT model, performance is stably high with few examples, but decreases when we use more than a couple of examples; for Llama, however, a higher number of examples increases performance. In addition to the number of few-shot examples, we must also consider the type (e.g., Huzaifah *et al.* 2024 observed the benefits of carefully selecting examples, with the ones most closely related to the input content being most useful in clarifying the task to the model). We therefore compared the use of random examples against the use of specifically chosen examples. One approach we followed was manufacturing our own examples that imitated the style of the dialogues but concentrated in a few sentences the main phenomena that could be challenging in our data; we call these examples 'ideal' examples. We also tested a similar approach where we instead selected the most representative examples from real data; we call these examples 'real'. Finally, we also tested an approach where we used real examples from the data but we did not manually select the best; instead, we used sentence similarity[11] to adapt what the best were for each case. We separated a pool of dialogue turns; before inference, each input turn was compared against the pool of turns and the top n most similar turns were retrieved as few-shot examples; we call this approach 'Similar'. Table 8 shows a comparison of the performance under the different types of few-shot examples. Again, using the development data limits the interpretability of the results, but they do suggest that a careful selection of the few-shot examples is far from irrelevant.

Model	FS	Base prompt F1	Chatty prompt 1 F1	Chatty prompt 2 F1
GPT 4o mini	2	0.79	0.56	0.70
GPT 4o mini	3	0.80	0.59	0.72
GPT 4o mini	6	0.81	0.57	0.76
Llama 3 8B	1	0.29	0.11	0.27
Llama 3 8B	2	0.45	0.33	0.28
Llama 3 8B	3	0.44	0.39	0.35
Llama 3 8B	10	0.47	0.47	0.51

Table 7: Performance over development data comparing two prompt styles, two small models, and three amounts of randomly-selected few-shot examples. The tests cover the whole development split (dialogue 032x028), excluding the 1-10 turns used for few-shot learning. FS refers to the number of few-shot examples. The F1 scores represent the value after post-processing the output.

Following the work by Manikantan *et al.* (2024), we also tested whether dividing the task into two simpler tasks improved performance. For the first task, we ask the model to only label the heads of mentions, expecting that this will reduce the burden of having to decide which determiners and modifiers to include — we leave that for the second task, where the heads are expanded into the full mention span. For the second step, as it relies purely on

[11] Martino Mensio's Github

Model	Type of FS	FS	F1pp
GPT4omini	Similar	3	0.81
L3-8b-v1	Similar	3	0.47
GPT4omini	Ideal	3	0.75
L3-8b-v1	Ideal	3	0.42
GPT4omini	3 ideal + n random	1	0.79
GPT4omini	3 ideal + n random	2	0.79
GPT4omini	3 ideal + n random	3	0.78
L3-8b-v1	3 ideal + n random	1	0.45
L3-8b-v1	3 ideal + n random	2	0.43
L3-8b-v1	3 ideal + n random	3	0.50
GPT4omini	Real	3	0.77
L3-8b-v1	Real	3	0.37
GPT4omini	3 ideal + n real	3	0.79
L3-8b-v1	3 ideal + n real	3	0.44

Table 8: Performance with different types and numbers of few-shot examples (FS). We present F1 scores after the output was post-processed (F1pp).

Model	Approach	FS	F1pp
GPT4o mini	2 prompts	3	0.67
GPT4o mini	1 prompt + SpaCy	3	0.63
GPT4o mini	1 prompt	3	0.80
Llama 3 8B	2 prompts	3	0.33
Llama 3 8B	1 prompt + SpaCy	3	0.37
Llama 3 8B	1 prompt	3	0.44
GPT4o mini	2 prompts	6	0.67
GPT4o mini	1 prompt + SpaCy	6	0.64
GPT4o mini	1 prompt	6	0.81
Llama 3 8B	2 prompts	6	0.36
Llama 3 8B	1 prompt + SpaCy	6	0.43
Llama 3 8B	1 prompt	6	0.49

Table 9: Performance comparison with one prompt or splitting the task into mention-head detection and mention span expansion (with a second prompt or with SpaCy). We present F1 scores after post-processing the output (F1pp).

identifying dependencies, we test it both with an LLM and with a simple script using SpaCy. Table 9 shows performance under these conditions, with a single-prompt approach for comparison. We can observe that the two-prompt approach never surpasses the single-prompt approach, and that using SpaCy to expand the heads does not improve performance but actually hinders it.

A.1 Prompts

Zero-shot prompt

You are an NLP parser specialised on extracting mentions from text. Your output is fed to a coreference resolution system and an entity linking system. Therefore, your output should respect format restrictions and not add any comments; if there are no mentions, just return the original text. You will be given a text and you should extract all the mentions in it. You should return the text with mention opening tags <M> and closing tags </M>.

Base prompt

You are an NLP parser specialised on extracting mentions from text. Your output is fed to a coreference resolution system and an entity linking system. Therefore, your output should respect format restrictions and not add any comments; if there are no mentions, just return the original text. You will be given a text and you should extract all the mentions in it. You should return the text with mention opening and closing tags as in the examples.

'Chatty' persona-based prompt

You are a university student pair-programming with a partner, who is also a university student. Your partner says something and you need to understand it to respond. To show that you've understood your partner, you need to label the things that your partner has mentioned, to later think about what each of those mentions refers to.

'Chatty' dialogue-based prompt

You are going to see a bit of text from a dialogue partner. Think of all the objects, concrete or abstract, that they mention in their text. Return the text with <M> </M >tags framing the objects. Return nothing else. Here are some examples of how you've done this before.

B Appendix: Mention classification labels

The guidelines provided to annotators include numerous images and examples, spanning 28 pages, so here we only provide a brief summary.

The annotation units are mentions, be it to ele-

ments in the dialogue (e.g., a previous word now referred to with a pronoun), in the context (e.g., the participants, programming concepts, elements of the code being created, etc.), or both. These mentions can be one or more words, and they may be split by punctuation; if the words are part of the same mention, we label them as one broad span that includes the words and the spaces (and possibly commas or apostrophes) between them. If we did not highlight the space in between the words as part of the annotation, that would create separate mentions. If there is a filled pause between parts of one mention, the pause can be included as part of the mention. We also include repetition within the same unit; for example, in many cases the speakers will repeat an article – we annotate them all as part of the mention. After labelling the mention spans, the annotators also classified the mentions according to their grammatical number and the linguistic categories summarised below:

- Pronoun - Personal
- Pronoun - Demonstrative
- Pronoun - Other
- NP - Definite
- NP - Indefinite
- NP - Meta (this category was used for mentions referring to words as abstract concepts, not the meaning represented by the word)
- NP - Read-dictate (this category was used for read-out or dictated code)
- Name
- Name variation (this category was used for variable names that were not reproduced exactly, e.g., when the 'len()' function is mentioned as 'length')
- Location adverb
- Incomplete (this label was added independently of the others to mentions that spanned more than one turn, so that we could later join the segments)