Discourse annotation — Towards a dialogue system for pair programming

Cecilia Domingo* — **Paul Piwek*** — **Svetlana Stoyanchev**** — **Michel Wermelinger***

* *The Open University, United Kingdom*
** *Toshiba Europe Limited, United Kingdom*

ABSTRACT. *Much work has been carried out on dialogue system development in different fields. With recent advances in Programming Language Processing tasks, dialogue systems aimed at programmers are becoming another viable area of application. However, the data necessary for a dialogue system that can assist programmers involves not only code, but the natural language around it. How should this data be annotated? In this review we examine the most common approaches to dialogue annotation, paying special attention to programming settings. We first look at the broader theories that inform these approaches, and after our review of the most widely used annotation schemes we analyze the peculiarities of the programming context and how well suited the existing schemes are for this setting.*

RÉSUMÉ. *Le développement de systèmes de dialogue a fait l'objet d'une grande attention dans différents domaines. Avec les progrès récents des tâches de traitement du langage de programmation, les systèmes de dialogue destinés aux programmeurs deviennent un autre domaine d'application viable. Cependant, afin de développer un système de dialogue pour assister les programmeurs, il est nécessaire de traiter non seulement le code, mais aussi le langage naturel associé. Comment ces données doivent-elles être annotées ? Dans cet article, nous présentons une synthèse des méthodes les plus courantes d'annotation des dialogues, avec un accent particulier sur le domaine de la programmation. On considère d'abord les théories sur lesquelles ces méthodes sont basées, on énumère les principales méthodes et on analyse les particularités du domaine de la programmation et dans quelle mesure les principales méthodes d'annotation sont adaptées à ce domaine.*

KEYWORDS: *dialogue systems, discourse annotation, programming language processing.*

MOTS-CLÉS : *systèmes de dialogue, annotation discursive, traitement du langage de programmation.*

1. Introduction

Dialogue systems have become pervasive in our everyday life, spanning a wide range of domains (Liu *et al.*, 2020; Kuyven *et al.*, 2018; Thoppilan *et al.*, 2022). While programmers put their efforts into developing this wide range of systems, few dialogue systems exist which focus on assisting these programmers. There exist other kinds of tools which assist programmers in some ways. Recently, for instance, Github released Copilot [1], a tool that autocompletes code, and Amazon released the similar CodeWhisperer tool [2]. In educational settings, there are numerous Intelligent Tutoring Systems (ITS) that can give feedback or some guidance to people learning to program (Keuning *et al.*, 2019). However, none of these tools follow a dialogic approach and harness the potential of dialogue systems.

One activity where programmers would highly benefit from a dialogue system assisting them is pair programming. Pair programming is a technique where two programmers work together on one piece of code (Hanks *et al.*, 2011). A sample from a pair-programming session can be found below in Example 1 [3]; we will be using it to illustrate different phenomena. In this session, two programmers demonstrate the technique, writing a program that detects high temperatures from an input list of temperatures. In this particular session, both participants are sitting together in front of the computer. However, only one of the participants takes control of the mouse and keyboard, assuming the role of "driver"; the other participant, the "navigator", collaborates with verbal guidance. The video allows us to see not only the participants, but also the screen where they are coding (albeit with faulty synchronization in this case) and a whiteboard they use for brainstorming, all in different windows merged into the same video. The technique they demonstrate, pair programming, is widely used, and has been proven very beneficial, especially in educational contexts (ibid). However, it can be difficult to implement, due to scheduling problems or partner incompatibility (ibid). A dialogue system could help ameliorate these issues. In fact, Wizard-of-Oz studies have already demonstrated the value of a dialogue system as a pair-programming partner: it could bring many of the benefits of pair programming (e.g., better performance, higher confidence), and it would be valued by users (Kuttal *et al.*, 2021). However, there is not sufficient data to develop such systems (Wood *et al.*, 2018).

Before more data can be made available, it is important to reflect on what this data would look like and how it would need to be processed. Some dialogue systems can be built with unannotated data (Thoppilan *et al.*, 2022). However, such systems require very large amounts of data and high computational power (ibid). A more viable alternative is to follow a supervised approach. Our envisioned system would enter the category of task-oriented systems, as it should collaborate with the human user to

1. https://github.com/features/copilot/.

2. https://aws.amazon.com/codewhisperer/.

3. We recommend watching the video of the session to understand the context of the sample. https://youtu.be/zdE2MS6gcbE.

achieve a goal, which in this case is developing a program. For such dialogue systems, the usual approach is a dialogue-state architecture (Jurafsky and Martin, 2021). These systems have a component that detects relevant entities in an utterance (slot filling), a dialogue state tracker that records the dialogue state at each point by considering the slots and the dialogue acts (which we shall discuss in Section 2.2.1), and a set of dialogue policies that determine the system's actions base don the dialogue state. These components require annotated data for training and testing the components — in a supervised approach, the target slots and dialogue acts need to be known. Then, exactly which annotations do we need for dialogues in order to develop a system to replace a human partner in pair-programming sessions? In this paper we reflect on this issue by analyzing existing theories of dialogue and the annotation schemes derived from them, and compare our findings with the characteristics of pair-programming dialogue.

Example 1 — Sample from pair-programming session

(1)	**Navigator:**	You could, you could put a print list if you want, just to.
(2)	**Driver:**	Okay.
(3)	**Navigator:**	But I would always just run it each time and I generally keep versions as well. I don't think we need to do that here, but I would keep... Could build version A, version B so that, if something goes wrong, you can get back to something that worked.
(4)	**Driver:**	[Typing] Right.
(5)	**Navigator:**	Yeah, again, it's just there, maybe there may be other ways of doing it now that I don't know, but I would just save a, b, c, 1, 2, 3.
(6)	**Driver:**	Okay. [Pointing at the screen] So that's our first version anyway. So saved that.
(7)	**Navigator:**	Just run it.
(8)	**Driver:**	[Overlapping] If we run...
(9)	**Navigator:**	It shouldn't, don't get any errors. Something works. [Looking at screen] Yeah, it works, fine.
(10)	**Driver:**	[Overlapping] There you go. Okay.
(11)	**Navigator:**	Fine. Okay. So where are we? [Looking at reference book] So we've got our input, [looking at whiteboard] so we're back to our pattern, right. So I know it needs a list. Set some sort of variable to the first item in the list.
(12)	**Driver:**	Uhum, okay. So again, uuum, what do we call it, something that's sensible again.
(13)	**Navigator:**	It's the highest temperature, isn't it the term?
(14)	**Driver:**	[Overlapping] [Typing] Aye.
(15)	**Navigator:**	Highest value or something. Put maybe highest temp, it's probably, since you've used temp at the...[Shrugging] Yeah, highest Temp.
(16)	**Driver:**	[Typing] And... we're gonna save that.
(17)	**Navigator:**	[Overlapping] The first item in the list
(18)	**Driver:**	[Typing and mumbling] Sensor...
(19)	**Navigator:**	[Mumbling] Temps...
(20)	**Driver:**	[Typing and mumbling, overlapping] Temps. And... okay.
(21)	**Navigator:**	Yeah.

(22) **Driver:** Wanna do zero?
(23) **Navigator:** Zero.
(24) **Driver:** Zero.
(25) **Navigator:** I think...
(26) **Driver:** Because...
(27) **Navigator:** Because an array always, well, doesn't always, that's the problem. And Python...
(28) **Driver:** [Overlapping, smiling] Python is...
(29) **Navigator:** An array starts...S list, an array. If I'm saying array...amm, a list...
(30) **Driver:** [Overlapping] Yeah
(31) **Navigator:** A list starts at zero.
(32) **Driver:** [Overlapping] Zero.

2. Discourse theories

2.1. *Definition*

Discourse can be defined as "joint activities in which conventional language plays a dominant role" (Clark, 2005, p. 50). This broad definition can be considered even broader if we take Skidmore's continuum of addressivity (Skidmore, 2019), where dialogues can range from the truly dialogic to the monologic. For the purposes of analyzing discourse in relation to dialogue systems, we wish to emphasize in this review the joint-activity aspect of discourse to obtain insights applicable to the more dialogic part of the spectrum (ibid). Thus, we will look at discourse theories as they apply to dialogue and not other types of discourse.

2.2. *Key concepts*

Numerous discourse theories have been developed, with more than a few achieving great influence. Therefore, instead of providing a detailed account of each of them, we will now summarize the key themes they cover.

2.2.1. *Acts and actions*

As we will discuss in more detail in Section 3.2, one of the theories that has had the strongest influence in how dialogue is conceptualized in NLP is Speech Act Theory (Austin, 2018). In this view, utterances are actions performed by the participants in discourse. The theory describes several levels of actions, from the phonetic act of making noises to the perlocutionary act of causing an effect on the hearer.

When the focus of the analysis is on the characteristics of discourse as an action between participants, instead of merely looking at the micro details of phonetics or syntax, we must turn to the level of illocutionary and perlocutionary acts and their effects. Austin (2018) distinguishes numerous types of such effects, such as verdictives (the

act of, as the name suggests, emitting a verdict) or exercitives (the act of issuing a recommendation, like (1) in our example). This classification was then built upon by Searle (1979) and several other authors.

Austin's idea of acts emphasizes the effects of utterances, yet it pays little attention to the interaction between the participants originating and receiving these effects. This cooperative element of the speech acts was explored by Clark and Schaefer (1989).

2.2.2. *Cooperative dimension*

The central view in Clark and Schaefer's (1989) theories is that using language is performing a joint action. Participating in dialogue is not seen as the sum of speakers' individual actions, but as a coordinated activity between them. Perhaps the most influential account of how participants cooperate in discourse are Grice's (1957) maxims: "make your contribution as informative as is required", "try to make your contribution one that is true", "be relevant", "be perspicuous" (Grice, 1991, p. 26). These are the rules that allow speakers to follow the Cooperative principle and achieve the goal of their discourse. The Cooperative principle states that speakers should make the contributions to the joint activity (discourse) that are required to achieve its goal (Grice, 1991). Sperber and Wilson (2010) build on Grice's work to develop their relevance theory. One key aspect of it is that communication relies on participants inferring meaning from the speaker's utterances. Inferences are linked to relevance: the inferred meaning of an utterance should be relevant to the context. Relevance, on its part, is mediated by effort: "an assumption is relevant in a context to the extent that the effort required to process it in this context is small" (*ibid* p. 125). Warren (2006) also builds on Grice's work, observing cooperation as a feature of naturalness in conversation data. A more dialogic equivalent of the Cooperative principle is the principle of least collaborative effort (Clark, 2005): participants will try to minimize the total effort of the joint activity, though this may involve putting additional effort on producing the utterances so that little effort is needed to understand them. For instance, in (5) of our example, the navigator essentially repeats what he says in utterance three, possibly trying to emphasize the goal of the utterances.

Gregoromichelaki *et al*. (2011), addressing some limitations in Grice's theories, discuss another factor that enables cooperation: incrementality. Discourse is produced and processed gradually, which enables participants to adjust it based on the feedback they receive. Yet the most widely discussed concept when studying the social elements of discourse is grounding, or how participants in discourse build a common ground (Clark and Schaefer, 1989; Clark, 2005).

2.2.3. *Common ground*

The common ground in a joint activity can be defined as the shared knowledge available to participants in discourse, be it knowledge about the world or the joint activity itself (Clark, 2005). Joint activities begin with an initial common ground that is built upon as the activity progresses, through accumulation or even deletion (Clark and Schaefer, 1989; Clark, 2005). Conceptualizations of the common ground can be built

as iterative propositions ad infinitum (i.e., it involves the speakers knowing that X is true, knowing that they know, knowing that they know they know, etc.); but actual processing cannot be expected to take place this way (ibid). In conversation, for the common ground to be built upon (grounding), participants need to provide sufficient evidence that they have adequately processed each other's contributions (ibid). This may be done through showing continued attention, making a new contribution that is relevant to their counterpart's, acknowledging understanding, repeating all or part of the contribution or displaying understanding some other way (ibid). While this is still a recursive process, with participants giving evidence of understanding contributions, then of understanding the understanding and so forth, this recursion does not continue ad infinitum, as the required level of evidence becomes weaker for each iteration (ibid). For instance, in our example the driver often acknowledges understanding of the navigator's utterances simply by saying "okay", and then they can move on to a different contribution. The concept of common ground is more concretely conceptualized by Grosz and Sidner (1986), who present the idea of a focus space, a discourse dimension containing the purpose of a discourse segment and the available referents for it. Pickering and Garrod (2004) adopt Clark's definition of the common ground, but argue that most dialogue uses a simpler, implicit common ground. This is built as speakers align their situational models; this alignment facilitates both comprehension and production for the speakers.

Beyond the theory, the concept of common ground has also been empirically tested. For instance, Jordan and Walker (2005) created a model to predict the content of referring expressions, and some of the features they employed reflected the theories on common ground (e.g., previously used attributes, other referents that could act as distractors, attribute saliency, etc.). Although the model's accuracy did not reach beyond 50%, the features related to the focus space showed some predictive power (features related to differences between the referent and distractors present in the focus space). Mitchell *et al*. (2012) also carried out some experiments regarding the common ground through the study of convergence (how participants in discourse adapt to each other). After analyzing tutor-mentee interactions over several weeks, they saw that lexical convergence increased over time: the words preferred by a participant became shared knowledge. An extensive practical study of convergence has also been carried out by Dubuisson *et al*. (2021), who developed a framework to compute measures of alignment and used it to analyze dialogues between humans and between humans and agents, observing differences in the flexibility of the alignment. Their work also opens doors for improving alignment in dialogue systems. Another important contribution that bridges theory and practice is Ginzburg's conversation theory, which builds a grammar for dialogue based on corpus data.

As we have mentioned, the focus space contains the referents and knowledge available and relevant to a particular discourse segment (Grosz and Sidner, 1986). In addition to that, it contains the purpose of the segment (ibid). In Grosz and Sidner's model, that intention that the segment tries to achieve determines how discourse is segmented: discourse units each have their own purpose contribution to the purpose of the overall

discourse (ibid). However, it is not only their theory which is primarily driven by the concept of purpose or intention – intention is another key pillar of discourse theories.

2.2.4. *Intention*

Grosz and Sidner (1986) see intentions as the element that structures discourse: the whole discourse will have a purpose (discourse purpose, or DP), but there will also be sub-goals that define the segments of the discourse (discourse segment purpose, or DSP) and become the key element of the focus space at any point. Even before this model, intention was already seen as a driving force within discourse. Austin (2018) reflected it through the illocutionary force of utterances: people say things to achieve a certain effect, and the type of desired effect is what allowed Austin and then Searle (1975, in Clark, 2005) and other authors to classify utterances. Grice (1991, p.26) also emphasized the importance of intentions through the Cooperative principle: "make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged". Grice (1957) also points out that it is not enough for utterances to have an intention, but that speakers must intend for this intention to be recognized, and that other participants in discourse must recognize the intention. As influential as Grice's theories have been, this and the theories it has guided have not been free of criticism. Gregoromichelaki *et al*. (2011), while accepting the concept of intention as useful for high-level conceptualizations of discourse, reject the claim that it has any significant effect in online processing of discourse due to its intractability. One of their arguments is the fact that people with low skills/experience in detecting other speakers' intentions and emotions (e.g., children and people who have autism but good verbal skills) may be able to participate in discourse (ibid). Instead, what would drive discourse processing is social conventions (ibid). When signals are used in non-conventional ways, there are other resources that can help participants solve the coordination problems of discourse, such as explicit agreement, saliency or precedents (Clark, 2005). For instance, in our example the speakers agree to refer to variables regarding temperature as "temp".

2.2.5. *Structures*

Finally, after having discussed the concepts on which discourse scholars define discourse structure, we can provide a brief account of what these structures look like. We have presented Clark and Schaefer's model (1989) as highly influential; this model, stemming from the idea that discourse is a type of joint action that uses language, views discourse as a tree of linked contributions and acceptances. One participant contributes an utterance, but for it to be added to the common ground and for participants to coordinate the individual actions into the joint action, the contribution needs to be accepted – and the acceptance utterance then becomes a new contribution for the first participant to accept.

Grosz and Sidner (1986) provide another model of discourse structure which accounts for more elements in it. As we mentioned, they establish purpose as the driving force

that defines the discourse segments. The segments can be nested, and dominance and precedence relations can be established between them. These segments with distinct purposes contributing to the overall discourse purpose form the intentional structure of discourse. They also define the attentional structure of discourse: this is a stack of focus spaces, each containing the purpose of the segment and the referents and relations relevant to it. The last structural element of discourse is the linguistic structure, the mere sequence of utterances.

Another type of structure that has been widely studied is rhetorical structure, especially through Rhetorical Structure Theory (RST) (Mann and Thompson, 1988, in Hou *et al*., 2020). This theory sees the structure of discourse as links between utterances building a coherent unit (Hobbs, 1979, in Moore *et al*., 2003). In RST, discourse units are schemas consisting of a nucleus and one or more satellites (ibid). The nuclei convey the core content of the discourse, while the satellites support the nuclei or other satellites through rhetorical relations like motivation, generalization, evidence, etc. (ibid). For instance, in (5) of our example, we can observe an antithesis relation between the first and second part. Although RST was initially devised for monologic discourse, it has later been applied to dialogue (Daradoumis, 1993).

2.3. *Links between discourse theories and NLP*

In the previous section we have presented some of the most influential discourse theories, and we have also briefly mentioned some empirical studies that drew from those theories. Still, as Pery-Woodley and Scott (2006) observed, there is some divide between discourse theories and NLP tasks that could benefit from them. Discourse theories help us conceptualize the macro-structure of discourse (e.g., cross-paragraph relations), but NLP tends to be more concerned with the micro-elements (e.g., the purpose of an individual sentence), as macro-structure is more intractable (ibid). On the other hand, discourse theories would also benefit from a link to empirical studies and NLP techniques that could test the theories and find connections between macro- and micro-structures. When it comes to bringing the insights from discourse theories into the NLP domain and, more specifically, to dialogue system development, we identify two key challenges: multimodality and online processing.

2.3.1. *Multimodality in dialogue*

Some of the dominating discourse theories acknowledge that dialogue involves both verbal and non-verbal signals (Clark and Schaefer, 1989; Grice, 1991; Clark, 2005). For instance, in (6) of our example, the driver uses both pointing gestures and demonstrative pronouns to refer to the program. For Clark (2005, p. 13), a signal is "any action by which someone means something for another person", and most combine modalities. Beyond theory, multimodality is a key concept in NLP research, and increasingly so (Admoni and Scassellati, 2014). Thus, we need to examine how truly suitable the theories are to account for the processing of non-verbal signals.

2.3.2. *Online processing*

In dialogue systems, the need for online (i.e., live) processing is undeniable, as "dialogues are created incrementally" (Skantze, 2021, p. 83): when we observe dialogue live, we cannot observe the final product, only its gradual development. Clark (2005, p. 29) already advocated for an "action approach" over a "product approach" to discourse. However, the main theories we have discussed fail to account for incrementality. Grosz and Sidner's model (1986), with the constant updating of the focus space stack, reflects this notion at the utterance level, but does not accurately represent incremental processing of utterance sub-elements (Gregoromichelaki *et al.*, 2011). Skantze (2021) aims to fill this gap with a model of incremental processing focused specifically on dialogue systems.

3. Annotation schemes

Annotation schemes help us bridge the gap between discourse theories and NLP tasks, as they facilitate discourse processing. Two of the main theories that are employed are Rhetorical Structure Theory (RST) and Speech Act Theory, each with a different focus. RST conceptualizes discourse through the relations between units. Although rhetorical relations represent speakers' intentions, a stronger emphasis is placed on the idea of intentions by Speech Act Theory. Another difference between the two theories is that RST emphasizes the relations between discourse units, whereas Speech Act Theory gives more weight to the actual discourse units. Given this divergence, we shall divide our account of annotation schemes into at least two trends.

3.1. *Relational schemes*

3.1.1. *Rhetorical Structure Theory (RST)*

RST conceptualizes discourse as consisting of propositions linked by coherence relations, which give rise to implicit propositions, helping us understand the initial propositions (ibid). In practice, it turns discourse into a set of schemas: a nucleus with satellites linked to it through rhetorical relations. For instance, in (5) of our example, the sentence starting with "but" would be a satellite linked to the rest of the utterance, the nucleus, through an antithesis relation.

Although RST was initially proposed for monologic discourse, it has later also been applied to dialogue. One approach has been to apply it to each turn separately, though this does not account for relations between utterances (Fawcett and Davies, 1992, in Daradoumis, 1993). Daradoumis (1993) addresses this by developing Dialogic Rhetorical Structure Theory (DRST), combining RST with an exchange model. The exchange model classifies RST schemas through additional dialogic relations: consent, elicitation, ascertainment. Within the schemas, it incorporates Clark and Schaefer's theory (Clark and Schaefer, 1989) by marking contribution and support relations. Another difference between DRST and RST is its dynamic nature: dialogue itself is

dynamic, so the representation is constantly changing, with schema nuclei and satellites changing their status (Daradoumis, 1993).

RST has successfully been harnessed for different NLP tasks, most notably summarization and text generation (Taboada and Mann, 2006; Hou *et al.*, 2020). Some research has also been done in dialogue: e.g., Fischer *et al*. (1994) design a dialogue system for database queries, where RST allows the system to represent the links between dialogue acts. NLP research using this theory has largely been enabled through the Rhetorical Structure Theory Discourse Treebank (RST-DT) (Carlson *et al.*, 2003). It is a corpus of 385 *Wall Street Journal* articles annotated by humans with 78 RST relations (ibid). Another key resource for NLP tasks involving discourse relations is the Penn Discourse Treebank (PDTB) (Prasad *et al.*, 2019), the largest resource of its type. Like the RST, it consists of annotated *Wall Street Journal* articles, but for the PDTB the number is 2,159 (Hou *et al.*, 2020).

Segmented Discourse Representation Theory (SDRT) SDRT was developed to address the shortcomings of RST and Discourse Representation Theory (DRT) (Lascarides and Asher, 2008). DRT (Kamp, 1981, in Asher & Lascarides, 2008) uses first-order logic to build semantic representations of discourse; combining these representations with RST reduces anaphora resolution options to the more "pragmatically preferred" (Lascarides and Asher, 2008, p. 1).

3.1.2. *Mapping relational annotation schemes*

With at least three very influential schemes for annotating discourse relations (RST, SDRT, PDTB), attempts have been made to map them for higher resource reusability. One approach is finding an intersection (Demberg *et al.*, 2019), or another is using an additional schema (Sanders *et al.*, 2021; Roze *et al.*, 2019). Bunt and Prasad (2016) proposed an ISO standard for coherence relation annotation; they also mapped it to the PDTB and RST.

3.2. *Speech Act Theory*

As discussed in Section 2.2, Speech Act Theory (Austin, 2018) sees utterances as the performance of speech acts. Acts range from the mere phonetic act of making sounds to the illocutionary act of uttering something with a purpose and the perlocutionary act of uttering something that has an effect (ibid). For instance, (15) of our example (Section 1) has the effect of the driver typing the variable name that the navigator suggested. The function of an utterance is called the illocutionary force, which can be implicit or can be made explicit through illocutionary force indicating devices (ifids) (Allan, 1997). These can be words like "please", the use of the imperative mood (as in utterance 15), or illocutionary verbs (verbs that perform an action by being uttered, such as "to name"). Austin focused on this last resource for the classification of illocutionary forces. Searle (1979) expanded upon Austin's classification and pointed out that speech acts can be implicit: the illocutionary force of an utterance

may be hidden behind the apparent performance of a different illocutionary act, such as when commands are softened by phrasing them as questions. Speech Act Theory has been critiqued on the basis that it focuses on illocutionary forces (what the speaker intends with an utterance), but these are not always evident: not to discourse annotators, and often even to hearers and speakers (Allan, 1997; Gregoromichelaki *et al.*, 2011). Another issue is that Speech Act Theory looks at speech acts after they are completed, whereas speakers process them incrementally in real time (ibid). Yet another critique is that Speech Act Theory looks at utterances as isolated acts (Allwood, 1977; Allan, 1997), without considering how they contribute to joint activities (Clark and Schaefer, 1989). Moreover, Austin and Searle's focus on illocutionary verbs places too much emphasis on the lexical dimension of discourse, neglecting others (Allwood, 1977). Despite all the criticism, Speech Act Theory has become highly influential for dialogue annotation, with numerous schemes based on it.

3.2.1. *Schemes derived from Speech Act Theory*

A large number of annotation schemes have been created which label speech acts (also known as "dialogue acts" in this context). For instance, Klein *et al*. (1999) already discussed sixteen influential schemes. Among the speech-act labelling schemes, the most widely used seem to be DAMSL and HCRC MapTask (either directly or through adapted versions).

DAMSL This scheme was developed by Allen and Core (1997) with the name Dialogue Act Markup in Several Layers. It has also been augmented for use with the Switchboard corpus (Jurafsky and Shriberg, 1997). As the name suggests, this scheme captures utterance intentions in several layers. The first layer, the forward communicative functions, concerns the speech acts (illocutionary acts). The other two layers are the backward communicative functions (how the utterance relates to the previous one), and the utterance features (form and content of the utterance). Unlike Speech Act Theory as developed by Austin (2018) and Searle (1979), DAMSL assigns several labels to an utterance (in addition to the scheme having three layers, each utterance can have several labels for each layer); this way it can capture the speech act with more nuance. This schema uses very generic labels with the goal of being applicable to any domain (e.g., statements are either assertions or reassertions; if not, they are simply "other"). However, this can result in some broad labels dominating in a corpus: for instance, Stolcke *et al*. (2000) annotated the general-domain Switchboard corpus and assigned the Statement label to more than a third of utterances. Weisser (2015) aimed to address this shortcoming of DAMSL by expanding it into a more nuanced scheme: DART. Whereas DAMSL has fewer than 15 functions per layer, DART has a total of 120 functions in its second version (Core and Allen, 1997; Weisser, 2015). Despite the large number of labels, it is intended to be simple to use thanks to its clear XML structure, which is aimed for easy automation and human interpretation (Weisser, 2015). DART has already been applied to a wide variety of dialogue types, such as political debates, call-center interactions or courtroom dialogue (ibid).

HCRC MapTask The HCRC MapTask project aimed to obtain dialogues that could be manipulated for some linguistic features (e.g., phonological characteristics of landmarks mentioned) (Anderson *et al.*, 1991). It collected data using a cooperative task design named the "map task", where two participants each have a map, but only one of them has information on the route that the other needs to follow on their map (Brown *et al.*, 1984). This approach to data gathering, which allowed for controlling numerous variables, as well as the corpus itself and the way it was annotated, have been highly influential. DAMSL, more closely linked to Speech Act Theory, focuses on the annotation of individual speech acts, paying only some limited attention to the relations between them. The annotation scheme of the HCRC MapTask, on the other hand, has three levels of annotations that go from the macro-structure of dialogue to the individual acts (Kowtko *et al.*, 1993; Carletta *et al.*, 1997). The highest level consists of transactions, which are sub-dialogues that serve to achieve the purpose of the overall dialogue. The middle level consists of dialogue games: they start with an initiation move and end when the goal of the game is achieved or abandoned. Finally, the lower level consists of the actual moves, a limited set of twelve speech acts. While the tagset of moves and games is widely used (Kowtko *et al.*, 1993; Chen and Di Eugenio, 2013; Ribeiro *et al.*, 2022), annotation at the level of transactions is very difficult, resulting in low inter-annotator agreement (Carletta *et al.*, 1997).

Adapted schemes It is difficult to find the right degree of granularity in a scheme so that the nuances of dialogue are accurately captured, but without making annotation too complex. For instance, one problem with studies that use versions of DAMSL is that a large number of utterances are labelled as "statement", so no distinction can be made between them (Margolis *et al.*, 2010; Robe *et al.*, 2020). The need for additional layers and tags is even more evident in tasks that involve more than textual data. Arguably, though, all NLP tasks could benefit from processing input in more than one modality: Stolcke *et al*. (2000), for instance, achieve greater speech act classification accuracy when combining word features with prosodic features. Some tasks involve yet more modalities, such as processing hand gestures. One example is the ELDERLY-AT-HOME multimodal corpus, where researchers had to expand the HCRC MapTask tagset to include labels that could describe the haptic actions involved in the task of providing care to elderly people — e.g., grabbing an object that the other person asks for (Chen and Di Eugenio, 2013).

3.3. *ISO standard*

As we have described, numerous annotation schemes exist, some focused on coherence relations, some more focused on speech acts, all with different degrees of granularity, and with varied suitability for multimodal annotation. In order to promote resource reusability, an ISO standard has been proposed for dialogue annotation, derived from the DIT++ annotation scheme (Bunt, 2009; Bunt, 2019; ISO, 2019). It appears to be mostly influenced by Speech Act Theory, as the focus is on annotating utterance function (ibid). It is intended to be suitable for multimodal annotation. Though

the annotation guidelines do not define any specific tags for non-textual modalities (e.g., eye gaze, facial gestures, hand movements, etc.), contributions in these modalities could be tagged with any of the standard's communicative function labels which described the function of the gesture. Furthermore, Petukhova and Bunt (2012) give some detailed examples of how such contributions could be coded following the ISO standard. In our example, for instance, the annotations for utterance 6 would feature a label called "handMove" with a shape parameter set to "pointing". The standard aims to be highly flexible: it consists of hierarchical labels across two general-domain functions and nine domain-specific functions (Bunt, 2009; Bunt, 2019; ISO, 2019). Depending on the needs of annotators, labels can be assigned on as many functions as is seen fit, and the hierarchy can be employed to the depth that is deemed appropriate (ibid). For instance, Zarisheva and Scheffler (2015) used the ISO standard on Twitter conversations; they attempted to simplify annotation by assigning labels on only one function per utterance, but found that this lowered inter-annotator agreement. The hierarchy also provides flexibility not only for how deeply or widely it can be used, but also for its expansion: the task function is given empty in the standard, leaving it to researchers to complete it with whichever functions they require (Bunt, 2009; Bunt, 2019; ISO, 2019). The ISO standard also defines a representation standard, the Dialogue Act Markup Language, which offers many options for enriching the annotations (ibid). For instance, it supports adding rhetorical relations through another ISO standard; thus, this standard can combine the two main trends in discourse annotation, with both speech act and rhetorical relation labels (ibid). In addition to the rhetorical relations, the standard also defines its own dependence relations between utterances: feedback and function relations (ibid). The standard has already demonstrated its usability for NLP tasks. For instance, Ribeiro *et al*. (2022) successfully apply convolutional neural networks to the classification of speech acts in dialogue text annotated with the standard.

3.4. *Segmentation*

We have discussed different ways in which dialogue segments can be labelled. However, the question remains of how to divide dialogues into relevant segments. In relational schemes, the common practice seems to be segmenting utterances by clauses (elementary discourse units) when using RST, and by sentences when using the PDTB scheme (Poláková *et al.*, 2017; Sanders *et al.*, 2021). When following a scheme based on Speech Act Theory, one option is annotating full turns (i.e., everything a speaker says before being interrupted by another, or before ending the dialogue) (Das and Pon-Barry, 2018). However, a common approach is annotating by utterances, which can be equivalent to a turn, but also to a set of turns or a turn fragment (Bunt, 2009; Bunt and Prasad, 2016; Bunt, 2019). What does then define the boundaries of an utterance? Core and Allen (1997), in line with Grosz and Sidner's (1986) theory of intentional structure, find these boundaries in the changes of function: when the speech act starts fulfilling a new function, that marks the start of a new segment. Nevertheless, Weisser (2015) acknowledges that this concept of "utterance" is often very vaguely defined;

"utterance" is then defined by Weisser as the smallest independent unit with semantic and pragmatic content. Nonetheless, most works on dialogue annotation fail to provide an account of how segmentation was carried out, leaving the definition of utterance and segment vague and/or implicit.

3.5. *Processing dialogue without annotating discourse*

Discourse theories have been highly influential for the creation of annotation schemes to build the corpora used for developing dialogue systems. Nonetheless, such systems can also be built without complex discourse data. In fact, Pery-Woodley and Scott (2006) point out that the macro-structures of discourse are built through processing micro-structures, but macro-structures are not linguistically explicit, which is one of the main issues for discourse processing in NLP. An option then is to focus on the smallest micro-elements. This is viable, for instance, in medical dialogue systems: Liu *et al*. (2020) build a dialogue system that identifies patients' gastrointestinal problems based on the recognition of entities (specific words describing symptoms and medicines). Advances in deep learning and the higher computational capacity that has enabled these advances also offer the alternative of developing dialogue systems without annotating discourse. Recently, for example, Thoppilan *et al*. (2022) trained transformer models with billions of unannotated dialogues and other texts to develop a generic dialogue system. The models were then fine-tuned with data obtained by having crowd-workers interact with the system (ibid).

4. Pair-programming discourse

As we have detailed in the previous section, there are numerous schemes available for dialogue annotation. Some of them are focused on specific tasks, whereas others aim to be applicable to a wide range of dialogue types; some analyze discourse through the relations between segments, whereas others look at the speaker's intention when uttering the segments. And there is also the ISO standard, which combines the different approaches and claims to be suitable for a wide range of dialogue annotation tasks (Bunt, 2009; Bunt, 2019; ISO, 2019). Are this and other schemes then well-suited for the annotation of dialogues produced in programming tasks, or how would they need to be customized? To answer this question, we first need to look at what distinguishes dialogue in this context from other types of dialogue.

We can observe several different types of interactions among programmers – there are even instances of programmers talking to a cardboard image of another programmer to develop their ideas (Bryant *et al.*, 2008). We would like to focus on pair-programming interactions, as this is a widely employed and studied practice which has proven to be very beneficial (Hanks *et al.*, 2011). It has also been shown that a dialogue system facilitating this type of interaction would be highly valuable (Robe *et al.*, 2020; Kuttal *et al.*, 2021; Robe, 2021). However, no such tool exists yet – annotated pair-programming dialogue would bring its development closer to reality.

Such data is lacking; in fact, even unannotated pair-programming data is scarce. Most works on pair programming are primarily concerned with the results of the practice, so researchers perform their analysis on the code produced or on participant surveys rather than dialogue samples (Werner and Denning, 2009; Hanks *et al.*, 2011; Adeliyi *et al.*, 2021). There are some exceptions; below we summarize the main characteristics of pair-programming dialogue that we have been able to extract from the literature. As we will detail in Section 5.1, we later aim to gather our own data to enrich the pool of knowledge on this type of dialogue. In Section 1, we already provided a brief definition of pair programming, as well as a sample transcription from a session. It is part of the Agile approach to software development; it is "a technique in which two individuals share a single computer as they work together to develop software" (Hanks *et al.*, 2011). The programmers normally take either of two roles: the driver and the navigator – though there is some debate about whether these roles are so clearly distinct or even always present (Hanks *et al.*, 2011, p. 135). The driver is the person in charge of the mouse and keyboard, who writes the code, while the navigator offers some guidance. These roles are often switched. Below we detail what the roles imply for the dialogue, as well as other characteristics of pair-programming dialogue. We then end this section with a brief illustration of the phenomena that are observable in the sample we used as an example.

4.1. *Roles*

Some of the literature on pair programming, especially when offering guidelines for the practice, defines very distinct navigator and driver roles, and recommends switching frequently. However, there often is no observable difference in how navigator and driver talk (Bryant *et al.*, 2008) with regard to levels of abstraction; the main difference is mainly control of the keyboard, which may affect how decisions are made, unless both participants have exactly equal access to the keyboard – such as when there are two keyboards. Still, differences may be observed in the amount of talk, as the driver will be focused on typing. Below we summarize how the roles affect some features of the dialogue.

Switching roles: Switching frequency might be imposed, especially in educational settings. However, students may find this hinders the natural workflow (Tsompanoudi *et al.*, 2013). On the other hand, in some sessions the roles are fixed, particularly if participants do not have two keyboards to switch easily. When switches happen, they're often taking advantage of pauses; sometimes, though, the navigator requests to switch. Sometimes it is easier for the navigator to switch and type than to explain what they mean (Chong and Hurlbutt, 2007; Zarb and Hughes, 2015).

Driver "muttering": The drivers, while typing, may verbalize what they are typing; as most of their attention may be devoted to the typing, their verbalization may be in the form of "muttering" (Zarb and Hughes, 2015). Whatever the form of the verbalization, it is encouraged by experts, as it allows the navigator to understand what the driver is thinking and see whether they need assistance (ibid).

Navigator giving instructions: Navigators are encouraged to offer suggestions to contribute to the program (ibid). In less equitable interactions, these suggestions may be given in a more domineering tone (Lewis and Shah, 2015). However, such commands without justification hinder collaboration (Wegerif and Mercer, 1996).

Driver deciding unilaterally: Another example of inequitable interaction and, thus, unsuccessful collaboration, is when the driver makes decisions on their own. When only one keyboard and mouse is available for pair programming, whoever has control of it has the final say about what makes it into the code (Chong and Hurlbutt, 2007). For minor decisions, it may be easier for the navigator to accept what the driver chose instead of arguing (ibid).

4.2. *Multimodality*

Like any other form of spoken dialogue, pair programming involves more than verbal communication (Clark, 2005). Additionally, as pair programming involves working on some code, the code will be commonly referenced and thus become a key element of the discourse.

Spoken discourse: The fact that we are dealing with spoken dialogue has some implications for its structure. The goal of an utterance may not be clear to the speaker from the beginning (Gregoromichelaki *et al.*, 2011), and this may be reflected in the structure: sentences may be ungrammatical, there may be repetitions, speakers may stutter, etc. This is also true for the overall structure of the discourse: regardless of each individual speaker's intentions, the intentional structure of the discourse arises from the joint dialogue (Grosz and Sidner, 1986). Spoken discourse also introduces prosodic features as an important modality to analyze. In unstructured discourse like a natural pair-programming interaction, utterances may be as simple and ambiguous as "mmmm"; a sound like this can indicate both approval and disapproval depending on the tone (Zarb and Hughes, 2015). Prosody also helps speakers manage turns (Skantze, 2021).

Gestures: Facial expressions and other body movements play an important role in communication. For instance, eye gaze can signal attention and serve as a turn-management device (Heylen *et al.*, 2002). A special kind of gesture is pointing (Chen and Di Eugenio, 2013); this gesture allows speakers to call a referent into the discourse. In the case of pair programming, this can be reference materials (a book, a whiteboard, a digital guide, etc.) or an element of the code. Additionally, pair programming has the peculiarity that pointing can be performed with either the body or an input device.

Actions replacing utterances: Non-verbal actions can contribute to the joint activity just like linguistic actions (Clark, 2005). The driver coding as a reaction to a suggestion from the navigator is a form of uptake (ibid). Also, sometimes the navigator may find it easier to request to drive and type in code than having to describe it (Chong and Hurlbutt, 2007).

4.3. *Skill levels*

When the participants have different skills levels, that has an impact on the collaboration (Chong and Hurlbutt, 2007; Plonka *et al.*, 2015). Due to the interaction of diverse factors, pairing programmers with different levels may result in both equitable and inequitable collaboration (Lewis and Shah, 2015). The goal of the session must be borne in mind: it may be knowledge transfer between expert and novice, or it may be simply to finish a project (Chong and Hurlbutt, 2007). Where time pressure is more important than the need to learn, the expert may take a more dominant role, giving direct instructions or driving without giving explanations (ibid). The novice, on their part, may take a passive role to avoid feeling like they are slowing down project completion or making a fool of themselves in front of the expert (ibid). When priority is given to knowledge transfer, an expert navigator may use mentoring strategies ranging from least to most explicit: hinting at problems, pointing out specific problems, and giving clear explanations (Plonka *et al.*, 2015). When the expert is driving, they can still mentor the novice by verbalizing their thoughts (ibid).

4.4. *Domain*

Pair programming is a specialized task and, as such, requires specialized terminology. This can come both from the programming domain and from the domain of the real-world problem that the code aims to solve. The combination of these domains also means that the content of the dialogue will switch among different "levels of abstraction": from the concrete level of the programming language's syntax, to the abstract real-world problem, going through the intermediate level of discussing code sections (Bryant *et al.*, 2008). It has been hypothesized that each of the two pair-programming roles deal with different levels. However, studies have contradicted this hypothesis, showing that both participants speak and initiate dialogue segments at all levels of abstraction (Chong and Hurlbutt, 2007; Bryant *et al.*, 2008) (as measured through specific coding schemes that tag these labels). The studies have also shown that most talk occurs at the intermediate level of abstraction, the discussion of code sections (ibid). Dialogue may also occur outside any of the levels, for instance, when the programmers deviate from the task and talk socially (Bryant *et al.*, 2008). However, these off-topic utterances may also be valuable for the interaction: e.g., they may help programmers disconnect from a difficult problem and reapproach it with a fresh perspective (Zarb and Hughes, 2015). Other disruptions to the workflow, of course, may be undesirable interruptions. One such kind may be an interruption from a third party outside the pair, or the pair programmers getting distracted (Chong and Siino, 2006).

4.5. *Collaboration*

Pair programming is a collaborative task, as participants work towards solving the same task jointly. For collaboration to be successful, participants need to discuss the

task with each other, make joint decisions, build on each other's ideas, correct each other's mistakes, and justify their contributions (Wegerif and Mercer, 1996; Bigman *et al.*, 2021). These features can be observed in successful implementations of the pair-programming technique. A good understanding between pair-programming partners is known as "jelling" (Adeliyi *et al.*, 2021). Jelling can result in some peculiar discourse phenomena. For example, participants may repeat what the other said to show uptake. Other phenomena resulting from good rapport may make the dialogue difficult to interpret to third parties. For instance, when there is great common ground, it may not be necessary for participants to finish their sentences: they may be finished by their partner or be left unfinished.

4.6. *Example*

In the example that we have presented (Section 1), we can observe many of the features that we have discussed. The participants only had one keyboard, which may have deterred them from switching roles. They have a similar skill level, which may have facilitated their successful collaboration. They establish such a good connection that they are even able to finish each other's thoughts, evidence of great shared common ground (Clark, 2005): e.g., when they discuss array characteristics in Python. Despite the equitable collaboration, we are able to see some of the characteristics that distinguish the role of driver and navigator: e.g., the navigator offers suggestions, and the driver types them in, in one instance verbalizing what they are typing through "muttering". The driver's actions also remind us of the importance of multimodality: non-linguistic actions such as saving a file when the navigator suggests doing so is a form of uptake. Multimodality is also important for turn-taking: we can clearly see the participants facing each other to indicate the end of a turn, especially after questions. They also often point at the screen or reference materials when discussing them. We also see several features reflective of spoken discourse. For instance, some sentences are left unfinished (e.g., "I think..."). We also see the navigator starting to discuss how arrays always start a certain way, only to change his mind mid-sentence (turn 27). Another interesting feature is the large number of sentences starting with "so" (e.g., turns 7, 11, or 12), structuring the discourse in an improvised manner. In this fragment we cannot see the whole range of abstraction levels that can be discussed in a pair-programming session — here we mainly see the intermediate level (discussing broad aspects of the program), which may be the most frequent one in pair programming (Chong and Hurlbutt, 2007; Bryant *et al.*, 2008). What we observe abundantly, though, is the use of terminology: the participants are often referring to temperatures (problem domain terminology), and using many programming terms (e.g., "print list", "variable", "array", etc.).

5. Conclusions

In the previous sections, we have looked at the most influential discourse theories and dialogue annotation schemes, as well as the main characteristics of dialogue during pair-programming sessions; thus, we may now attempt to answer our initial question: how can pair-programming dialogue be annotated for its analysis and the development of NLP-based systems that can assist programmers in these sessions?

Discourse theories, particularly Clark and Schaeffer's work (Clark and Schaefer, 1989), have allowed us to see that dialogues are joint activities. As such, they consist of contributions that need to be accepted by partners (uptake). Especially in a context like pair programming, where we seek effective collaboration, we need to annotate whether this uptake takes place. The ISO standard (Bunt, 2009; Bunt, 2019; ISO, 2019) captures this concept through dependence relations between utterances (feedback and functional relations), as well as feedback and dialogue management functions.

Grosz and Sidner's (1986) influential description of an attentional space and Clark's (2005) theories of common ground also show us the value of representing the speakers' shared basis. The ISO standard (Bunt, 2009; Bunt, 2019; ISO, 2019) and all the schemes based on Speech Act Theory allow us to annotate the purpose of segments; SDRT (Lascarides and Asher, 2008), on the other hand, allows us to build a semantic representation of the segments and find the most pragmatically plausible referents to resolve anaphora. With segment purposes and referents, we have the main elements of the focus space described by Grosz and Sidner (1986). As we mentioned in Section 3.1, SDRT combines DRT and RST. While the ISO standard does not include the logic representations of DRT, it does allow for annotation of the coherence relations of RST.

Another important aspect highlighted by influential scholars like Clark (2005), and evident also from observations of pair-programming sessions, is that multimodality is very important. The ISO standard (Bunt, 2009; Bunt, 2019; ISO, 2019) is designed to allow for the annotation of non-verbal contributions to discourse. The standard functions may not accurately describe all non-verbal contributions in pair programming, but the scheme allows for customization, especially through the Task dimension (ibid). For instance, we observed that sometimes a contribution may simply be the driver coding. Additionally, we observed pointing as a frequent form of non-verbal contribution. Chen and Di Eugenio (2013) adapt the HCRC MapTask scheme to add labels to describe pointing and other similar hand gestures; a similar Gesture dimension might be added to the ISO standard. The standard also allows for the optional annotation of three types of qualifiers: certainty, conditionality, and sentiment. These labels could allow annotators to capture the information conveyed through prosody and other non-verbal modalities.

We have mentioned the Task dimension of the ISO standard as an option for labelling task-related functions of non-verbal contributions, such as the driver coding. This dimension could code a lot of valuable information for the analysis of pair programming. For instance, Wood *et al.* (2018) distinguish utterances not simply by general

function, like Statement or Question, but also by programming-related topic (e.g., API or implementation). Other studies code stages of programming-related problem solving, such as "reviewing code", "muttering while typing", and "suggesting" (Zarb and Hughes, 2015). These studies also see the value of off-task contributions; less desirable interruptions of the workflow may also occur frequently, and as such might need to be coded to be differentiated (Plonka *et al.*, 2012). Another crucial concept related to pair programming is the participant's role. While it may not always be distinguishable from looking at the utterances (Chong and Hurlbutt, 2007; Bryant *et al.*, 2008), it would be useful to code who is using the keyboard and/or who has access to it. Roles are already annotated in some corpora in other settings, such as the HCRC MapTask corpus, which annotates the roles of Giver (the person giving directions) and Follower (the person following directions) (Anderson *et al.*, 1991).

Lastly, as the goal in pair programming is often for programmers to improve their skills through collaboration, it might be valuable to add labels stemming from pedagogical theories. For example, a very influential code for dialogue in collaborative learning tasks was developed by Wegerif and Mercer (1996). These authors distinguish one type of talk, disputational talk, which is not conducive to learning, as it merely fosters conflict. Tsan *et al*. (2021) offer a more fine-grained classification of conflict that includes some positive conflict. Task-related conflict may even constitute what Wegerif and Mercer (1996) consider the most effective kind of talk for learning: exploratory talk. Werner and Denning (2009) incorporate Wegerif and Mercer's theories with other codes inspired by Vygotskyan theories. Useful codes are also found in Plonka *et al*. (2015), who analyze mentoring strategies.

5.1. *Future work*

With this work, we aimed to draw some conclusions about how pair-programming dialogues could be annotated for the development of dialogue systems. To do this, we have extracted the most relevant insights from the literature. We started by looking at the theoretical work, to then examine more practical work on dialogue annotation and on the analysis of pair programming. This last section described our conclusions, linking the characteristics of dialogue in general, and pair-programming dialogue in particular, with existing annotation schemes. However, we have put special emphasis on the ISO standard. Standards can make linguistic resources easier to compare and reuse. The standard for dialogue annotation claims to be usable for a wide range of dialogue types and annotation uses, and it can be customized. Our analysis of the literature leads us to suggest that it might be suitable for the annotation of pair-programming dialogue, provided some customization is made and some of the optional annotations are used, as discussed above in Section 5. A Gesture dimension would need to be added, and the Task dimension would need to be developed with labels related to coding actions, pair-programming roles and features of collaboration. The optional sentiment qualifiers would be useful, and dependence relations should also be annotated.

Pair-programming dialogue and similar dialogues have been annotated with simpler schemes (Robe *et al.*, 2020; Kuttal *et al.*, 2021; Robe, 2021), but this may result in most of the dialogue being fit into one category. The imbalanced categories then make it difficult to train models that can be usable for NLP tasks, such as intent detection in dialogue systems (ibid): how can a dialogue system choose the appropriate policies if all it knows for most utterances is that they are statements, without any further distinction? The higher granularity that we recommend here, on the other hand, also has disadvantages. Firstly, complex annotation schemes may result in low annotation reliability. Additionally, the annotation process becomes slower and thus more expensive. Therefore, our next work will be testing our hypothesis that a customized version of the ISO standard is suitable for pair-programming data before any large corpus can be annotated. We have applied the insights from this paper to annotate the video from which we extracted our Example 1 (Section 1). This has allowed us to make some initial decisions about our annotation guidelines, which is the recommended procedure before annotating a larger corpus (Fuoli, 2018). Over the next months, we are planning to annotate additional videos that we have obtained; this shall help us better analyze pair-programming dialogue, as well as continue to refine our annotation scheme by introducing additional annotators. Afterwards, we plan to collect our own data to be able to gather richer multimodal input. Once our data is collected, we aim to start training and testing a subsystem of an NLU component for slot filling (see Section 1). It is our hope that the outcome of our research will then pave the way for the development of further system components and eventually the development of a dialogue system that can effectively function as a pair-programming partner.

* ACKNOWLEDGEMENT: This work has been carried out with financial support from EPSRC Training Grant DTP 2020-2021 Open University and Toshiba Europe Limited.

6. References

Adeliyi A., Wermelinger M., Kear K., Rosewell J., "Investigating Remote Pair Programming In Part-Time Distance Education", *United Kingdom and Ireland Computing Education Research conference*, ACM, Glasgow United Kingdom, p. 1-7, 2021.

Admoni H., Scassellati B., "Data-Driven Model of Nonverbal Behavior for Socially Assistive Human-Robot Interactions", *Proceedings of the 16th International Conference on Multimodal Interaction*, ACM, Istanbul Turkey, p. 196-199, 2014.

Allan K., "Speech Act Theory: An overview", *Concise encyclopedia of philosophy of language*, Pergamon, Exeter, UK, p. 454-467, 1997.

Allwood J., "A Critical look at Speech Act Theory", *Logic, Pragmatics and Grammar*, University of Göteborg, Lund, Sweden, p. 53-99, 1977.

Anderson A., Thompson H. S., Bader M., Bard E., Boyle E. H., Doherty-Sneddon G., Garrod S. C., Isard S. D., Kowtko J. C., McAllister J., Miller J., Sotillo C. F., Weinert R., "The HCRC Map Task Corpus: A Natural Spoken Dialogue Corpus", *Language and Speech*, vol. 34, n^{o} 4, p. 351-366, 1991.

Austin J. L., *How to do things with words: The William James Lectures deliveres at Harvard University in 1955*, Martino Fine Books, Eastford, CT, 2018.

Bigman M., Roy E., Garcia J., Suzara M., Wang K., Piech C., "PearProgram: A More Fruitful Approach to Pair Programming", *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ACM, Virtual Event USA, p. 900-906, 2021.

Brown G., Anderson A., Shillcock R., Yule G., *Teaching talk*, 1st edn, Cambridge University Press, 1984.

Bryant S., Romero P., du Boulay B., "Pair programming and the mysterious role of the navigator", *International Journal of Human-Computer Studies*, vol. 66, n° 7, p. 519-529, 2008.

Bunt H., "The DIT++ taxonomy for functional dialogue markup", *8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Budapest, Hungary, 2009.

Bunt H., *Guidelines for using ISO standard 24617-2*, Technical report, Tilburg University, 2019.

Bunt H., Prasad R., "ISO DR-Core (ISO 24617-8): Core Concepts for the Annotation of Discourse Relations", *ACL 2016*, 2016.

Carletta J., Isard A., Isard S., Kowtko J., Doherty-Sneddon G., Anderson A., "The Reliability of a Dialogue Structure Coding Scheme", *Computational Linguistics*, vol. 23, n° 1, p. 13-31, 1997.

Carlson L., Marcu D., Okurowski M. E., "Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory", *in* N. Ide, J. Véronis, J. van Kuppevelt, R. W. Smith (eds), *Current and New Directions in Discourse and Dialogue*, vol. 22, Springer Netherlands, Dordrecht, p. 85-112, 2003.

Chen L., Di Eugenio B., "Multimodality and Dialogue Act Classification in the RoboHelper Project", *SIGDIAL*, Metz, France, p. 183-192, 2013.

Chong J., Hurlbutt T., "The Social Dynamics of Pair Programming", *29th International Conference on Software Engineering (ICSE'07)*, IEEE, Minneapolis, MN, USA, p. 354-363, 2007.

Chong J., Siino R., "Interruptions on software teams: a comparison of paired and solo programmers", *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work — CSCW '06*, ACM Press, Banff, Alberta, Canada, p. 29, 2006.

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

Clark H. H., Schaefer E. F., "Contributing to Discourse", *Cognitive Science*, vol. 13, n° 2, p. 259-294, 1989.

Core M., Allen J., "Coding Dialogs with the DAMSL Annotation Scheme", *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, Cambridge, MA, p. 28-35, 1997.

Daradoumis T., "Towards a Representation of the Rhetorical Structure of Interrupted Exchanges", *Fourth European Workshop on Trends in Natural Language Generation, An Artificial Intelligence Perspective*, p. 106-124, 1993.

Das R., Pon-Barry H., "Turn-Taking Strategies for Human-Robot Peer-Learning Dialogue", *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, Association for Computational Linguistics, Melbourne, Australia, p. 119-129, 2018.

Demberg V., Scholman M. C., Asr F. T., "How compatible are our discourse annotation frameworks? Insights from mapping RST-DT and PDTB annotations", *Dialogue & Discourse*, vol. 10, n° 1, p. 87-135, 2019.

Dubuisson Duplessis G., Langlet C., Clavel C., Landragin F., "Towards alignment strategies in human-agent interactions based on measures of lexical repetitions", *Language Resources and Evaluation*, vol. 55, n° 2, p. 353-388, 2021.

Fischer M., Maier E., Stein A., "Generating Cooperative System Responses in Information Retrieval Dialogues", *Proceedings of the 7th International Workshop on Natural Language Generation*, Kennebunkport, Maine, 1994.

Fuoli M., "A stepwise method for annotating appraisal", *Functions of Language*, vol. 25, n° 2, p. 229-258, 2018.

Gregoromichelaki E., Kempson R., Purver M., Mills G. J., Cann R., Meyer-Viol W., Healey P. G. T., "Incrementality and intention-recognition in utterance processing", *Dialogue & Discourse*, vol. 2, n° 1, p. 199-233, 2011.

Grice P., "Meaning", *The Philosophical Review*, vol. 66, n° 3, p. 377-388, 1957.

Grice P., *Studies in the Way of Words*, 1991.

Grosz B., Sidner C., "Attention, intentions, and the structure of discourse", *Computational Linguistics*, vol. 12, n° 3, p. 175-204, 1986.

Hanks B., Fitzgerald S., McCauley R., Murphy L., Zander C., "Pair programming in education: a literature review", *Computer Science Education*, vol. 21, n° 2, p. 135-173, 2011.

Heylen D., van Es I., Nijholt A., van Dijk B., "Experimenting with the Gaze of a Conversational Agent", *International CLASS Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, Copenhagen, Denmark, p. 93-100, 2002.

Hou S., Zhang S., Fei C., "Rhetorical structure theory: A comprehensive review of theory, parsing methods and applications", *Expert Systems with Applications*, vol. 157, p. 113421, 2020.

ISO, ISO/DIS 24617-2, Second Edition, Technical report, 2019.

Jordan P. W., Walker M. A., "Learning Content Selection Rules for Generating Object Descriptions in Dialogue", *Journal of Artificial Intelligence Research*, vol. 24, p. 157-194, 2005.

Jurafsky D., Martin J., "Chatbots & Dialogue Systems", *Speech and Language Processing*, Stanford University, p. 1-39, 2021.

Jurafsky D., Shriberg E., "Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual", `https://web.stanford.edu/~jurafsky/ws97/manual.august1.html`, 1997. Accessed on March 10th, 2023.

Keuning H., Jeuring J., Heeren B., "A Systematic Literature Review of Automated Feedback Generation for Programming Exercises", *ACM Transactions on Computing Education*, vol. 19, n° 1, p. 1-43, 2019.

Klein M., "An Overview of the State of the Art of Coding Schemes for Dialogue Act Annotation", *in* G. Goos, J. Hartmanis, J. van Leeuwen, V. Matousek, P. Mautner, J. Ocelíková, P. Sojka (eds), *Text, Speech and Dialogue*, vol. 1692, Springer Berlin Heidelberg, Berlin, Heidelberg, p. 274-279, 1999.

Kowtko J. C., Isard S. D., Doherty-Sneddon G., "Conversational Games Within Dialogue", *HCRC Technical Report*, vol. 31, p. 1-12, 1993.

Kuttal S. K., Ong B., Kwasny K., Robe P., "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*, ACM, Yokohama Japan, p. 1-20, 2021.

Kuyven N. L., André Antunes C., João de Barros Vanzin V., Luis Tavares da Silva J., Loureiro Krassmann A., Margarida Rockenbach Tarouco L., "Chatbots na educação: uma Revisão Sistemática da Literatura", *RENOTE*, 2018.

Lascarides A., Asher N., "Segmented Discourse Representation Theory: Dynamic Semantics With Discourse Structure", *in* H. Bunt, R. Muskens (eds), *Computing Meaning*, Springer Netherlands, Dordrecht, p. 87-124, 2008.

Lewis C. M., Shah N., "How Equity and Inequity Can Emerge in Pair Programming", *Proceedings of the 11th annual International Conference on International Computing Education Research*, ACM, Omaha Nebraska USA, p. 41-50, 2015.

Liu W., Tang J., Qin J., Xu L., Li Z., Liang X., "MedDG: A Large-scale Medical Consultation Dataset for Building Medical Dialogue System", *arXiv:2010.07497 [cs]*, 2020.

Margolis A., Livescu K., Ostendorf M., "Domain Adaptation with Unlabeled Data for Dialog Act Tagging", *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, Association for Computational Linguistics, Uppsala, Sweden, p. 45-52, 2010.

Mitchell C., Boyer K. E., Lester J., "From strangers to partners: examining convergence within a longitudinal study of task-oriented dialogue", *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Seoul, South Korea, p. 94-98, 2012.

Petukhova V., Bunt H., "The coding and annotation of multimodal dialogue acts", *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, European Language Resources Association (ELRA), Istanbul, Turkey, p. 1293-1300, 2012.

Pickering M. J., Garrod S., "Toward a mechanistic psychology of dialogue", 2004.

Plonka L., Sharp H., van der Linden J., "Disengagement in pair programming: Does it matter?", *2012 34th International Conference on Software Engineering (ICSE)*, IEEE, Zurich, p. 496-506, 2012.

Plonka L., Sharp H., van der Linden J., Dittrich Y., "Knowledge transfer in pair programming: An in-depth analysis", *International Journal of Human-Computer Studies*, vol. 73, p. 66-78, 2015.

Poláková L., Mírovský J., Synková P., "Signalling Implicit Relations: A PDTB - RST Comparison", *Dialogue & Discourse*, vol. 8, nº 2, p. 225-248, 2017.

Prasad R., Webber B., Lee A., Joshi A., "Penn Discourse Treebank Version 3.0", `https://catalog.ldc.upenn.edu/LDC2019T05`, 2019. Accessed on March 10th, 2023.

Péry-Woodley M.-P., Scott D. R., "Computational Approaches to Discourse and Document Processing", *Trait. Autom. des Langues*, vol. 47, p. 7-19, 2006.

Ribeiro E., Ribeiro R., Martins de Matos D., "Automatic Recognition of the General-Purpose Communicative Functions Defined by the ISO 24617-2 Standard for Dialog Act Annotation", *Journal of Artificial Intelligence Research*, 2022.

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

Robe P., Kaur Kuttal S., Zhang Y., Bellamy R., "Can Machine Learning Facilitate Remote Pair Programming? Challenges, Insights & Implications", *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, IEEE, Dunedin, New Zealand, p. 1-11, 2020.

Roze C., Braud C., Muller P., "Which aspects of discourse relations are hard to learn? Primitive decomposition for discourse relation classification", *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, Association for Computational Linguistics, Stockholm, Sweden, p. 432-441, 2019.

Sanders T. J., Demberg V., Hoek J., Scholman M. C., Asr F. T., Zufferey S., Evers-Vermeul J., "Unifying dimensions in coherence relations: How various annotation frameworks are related", *Corpus Linguistics and Linguistic Theory*, vol. 17, n° 1, p. 1-71, 2021.

Searle J. R., "A taxonomy of illocutionary acts", *Expression and Meaning: Studies in the Theory of Speech Acts*, Cambridge University Press, p. 1-29, 1979.

Skantze G., "Turn-taking in Conversational Systems and Human-Robot Interaction: A Review", *Computer Speech & Language*, vol. 67, p. 101178, 2021.

Skidmore D., "Dialogism and education", *The Routledge International Handbook of Research on Dialogic Education*, p. 27-37, 2019.

Sperber D., Wilson D., *Relevance: communication and cognition*, 2nd edn, Blackwell, 2010.

Stolcke A., Ries K., Coccaro N., Shriberg E., Bates R., Jurafsky D., Taylor P., Martin R., Ess-Dykema C. V., Meteer M., "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech", *Computational Linguistics*, vol. 26, n° 3, p. 339-373, 2000.

Taboada M., Mann W. C., "Applications of Rhetorical Structure Theory", *Discourse Studies*, vol. 8, n° 4, p. 567-588, 2006.

Thoppilan R., De Freitas D., Hall J., et al, "LaMDA: Language Models for Dialog Applications", *arXiv:2201.08239 [cs]*, 2022.

Tsan J., Vandenberg J., Zakaria Z., Boulden D. C., Lynch C., Wiebe E., Boyer K. E., "Collaborative Dialogue and Types of Conflict: An Analysis of Pair Programming Interactions between Upper Elementary Students", *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ACM, Virtual Event USA, p. 1184-1190, 2021.

Tsompanoudi D., Satratzemi M., Xinogalos S., "Exploring the effects of collaboration scripts embedded in a distributed pair programming system", *Proceedings of the 18th ACM conference on Innovation and technology in computer science education — ITiCSE '13*, ACM Press, Canterbury, England, UK, p. 225, 2013.

Warren M., *Features of naturalness in conversation*, J. Benjamins, 2006.

Wegerif R., Mercer N., "Computers and Reasoning Through Talk in the Classroom", *Language and Education*, vol. 10, n° 1, p. 47-64, 1996.

Weisser M., "Speech act annotation", *in* K. Aijmer, C. Rühlemann (eds), *Corpus Pragmatics*, Cambridge University Press, Cambridge, p. 84-114, 2015.

Werner L., Denning J., "Pair Programming in Middle School: What Does It Look Like?", *Journal of Research on Technology in Education*, vol. 42, n° 1, p. 29-49, 2009.

Wood A., Rodeghero P., Armaly A., McMillan C., "Detecting Speech Act Types in Developer Question/Answer Conversations During Bug Repair", `http://arxiv.org/abs/1806.05130`, 2018. Accessed on March 10th, 2023.

Zarb M., Hughes J., "Breaking the communication barrier: guidelines to aid communication within pair programming", *Computer Science Education*, vol. 25, n° 2, p. 120-151, 2015.

Zarisheva E., Scheffler T., "Dialog Act Annotation for Twitter Conversations", *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Association for Computational Linguistics, Prague, Czech Republic, p. 114-123, 2015.