Can Machine Learning Facilitate Remote Pair Programming? Challenges, Insights & Implications

Peter Robe
University of Tulsa
Tulsa, OK, United States
pjr144@utulsa.edu

Sandeep Kaur Kuttal
University of Tulsa
Tulsa, OK, United States
sandeep-kuttal@utulsa.edu

Yunfeng Zhang
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
zhangyun@us.ibm.com

Rachel Bellamy
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
rachel@us.ibm.com

***Abstract*—Remote pair programming encapsulates the benefits of well-researched (co-located) pair programming. However, its effectiveness is hindered by challenges including pair incompatibility, imbalanced roles, and inclinations to work alone. Recent research has explored pedagogical methods to alleviate these challenges, but none have considered the integration of machine learning agents to facilitate remote pair programming. Therefore, we investigated the capabilities of popular text classification algorithms on identifying three facets of pair programming: dialogue acts, creativity stages, and pair programming roles. We collected a dataset of 3,436 utterances from a lab study of 18 pair programmers in a simulated remote environment. We found that pair programming dialogue poses a challenge as it is often unpremeditated and inadequately structured. Despite this, the accuracy of our machine learning classifier was improved by the choice of contextual dialogue features. Our results have implications for facilitating pair programming in global software development and online computer science education.**

I. Introduction

Remote pair programming (RPP) is a practice where two geographically separated programmers develop software together. The two programmers regularly switch between the roles of driver (writes the code) and navigator (reviews the code) [1], [2]. RPP inherits the merits of co-located pair programming, such as improving design and code quality, reducing errors, narrowing knowledge gaps, increasing confidence in the code, helping learn new languages and tools, and improving communication [2]–[24].

RPP also inherits the challenges of co-located pair programming, including pair incompatibility (e.g., differing skill levels, conflicting personalities, and imbalanced power dynamics) [25], [26], imbalanced roles (e.g., not sharing the driver role) [27], and inclinations to work alone to avoid being "slowed down" by their partner [26], [27]. Additionally the remote nature of RPP exacerbates challenges related to coordination, collaboration, and communication [23]. Although current research has explored various pedagogical methods to alleviate the challenges of co-located pair programming [28]–[31], we are the first to explore the viability of an RPP facilitator agent.

Group facilitator agents are a subset of Group Decision Support Systems (GDSS), which have a long history of supporting group decision making [32]–[41]. These systems help in different ways, such as de-emphasizing personal relations, equalizing member participation, making processes clearer, resolving conflict, and simulating the exploration of alternative ideas [33]. An RPP facilitator agent could apply these benefits in an RPP environment. A specific component of such an agent is the ability to classify dialogue. Therefore, it is necessary to investigate the potential utility of machine learning (ML) algorithms by classifying various facets of programmers' conversations and analyzing the resulting accuracies.

To guide detection and classification, ML algorithms rely on samples of dialogue. However, no pre-existing corpus of RPP dialogue exists, and dialogue collected for one domain is typically not applicable for use in another [42]. Additionally, there are stylistic differences between normal dialogue vs. pair programming dialogue that necessitate an alternative strategy to feature selection. Thus, the differing results and accuracy measures must be analyzed in regard to the limited dataset and unique feature selection strategy.

To understand the capabilities of popular text classification algorithms and to identify the most influential features in the domain of RPP, we formulated the following research questions:

- *RQ1: How effective are ML algorithms for classifying dialogue acts during RPP?*
  To measure the ability of ML algorithms to understand dialogue, we were interested in dialogue act classification. Dialogue acts are used by conversational agents as a means to identify the intent of the speaker [43]–[47].
- *RQ2: How effective are ML algorithms for classifying creativity stages during RPP?*
  To promote creativity among programmers, we were interested in classifying creativity stages. Creativity is highly desirable for an individual's success [48]–[50] and for solving open-ended problems [51], [52].
- *RQ3: How effective are ML algorithms for classifying RPP roles?*
  To promote coordination within pairs, we were interested in automatically classifying pair programming roles. This knowledge can be used to encourage equal participation by discouraging any participant from solely controlling the driver role [27].

978-1-7281-6901-9/20/$31.00 ©2020 IEEE

We conducted a lab study to collect a dataset of RPP conversations. Across nine sessions, 18 participants pair programmed in a simulated remote environment, totaling 3,436 utterances (a continuous piece of speech beginning and ending in a pause). Then, we manually labeled utterances to train a supervised ML classifier to automatically predict these labels.

II. Background

A. Dialogue Acts

A dialogue act (DA) represents the function of an utterance in dialogue (e.g., whether it is a question, request, command, offer, etc.). DAs are a type of speech act [53], [54] adopted from a philosophical origin and adapted to the empirical domain of computational modeling. While it often goes unnoticed, the subconscious mind passively processes DAs: detecting when a DA occurs, determining the type of DA, and formulating an appropriate response [55]–[58]. Likewise, a computer agent can be trained to perform the same classification procedure. In computer science, DAs are collected to uniquely cover the breadth of actions taken in a particular domain and are "based on pragmatic, syntactic and semantic criteria [43]." They are used to understand the intent of the speaker in the development of conversational agents [43]–[47] and tutorial dialogue systems [44]. Rodriguez et. al. [59] used DAs to study conversational speech in co-located pair programming. We adapted our DAs from [43], [44], [59] (refer Table I).

B. Creative Problem Solving

The Osborn-Parnes Creative Problem Solving Process is frequently used to understand the creative process of an individual [60]–[63]. It consists of four stages, as seen in Table I. Information, goals, and challenges are analyzed in the `Clarify` stage; programmers utilize divergent and convergent thinking in the `Idea` and `Develop` stages; and apply the resulting solutions in the `Implement` stage. The Osborn-Parnes creativity problem solving process provides a structured approach to solving open-ended problems, and has been used to teach programming in computer science education [64], [65].

C. RPP Roles

In pair programming, the prominent roles are `Driver` (controls the keyboard and writes the code) and `Navigator` (observes, guides the session, gives ideas, and reviews the code) [1], [2]. Research on pair interaction has shown that participants work together and discuss at the same strategic level of abstraction [66]. Therefore we added the `Discussing` role for when both participants conversed and neither controlled the keyboard.

TABLE I
Labels used for dialogue acts, creativity stages and RPP roles.

Dialogue Acts	**Definitions**
ABD (Abandoned)	An unfinished remark
ACK (Acknowledgement)	Acceptance of the existence of something
AN (Answer No)	"No" responses
APG (Apology)	A regretful acknowledgment of failure
AWH (Answer What How)	Answering who/what/when/where/why/how questions
AY (Answer Yes)	"Yes" responses
COR (Correction)	Verbally correcting
DIR (Direct Instruction)	An explicit instruction
FNON (Feedback Non-Positive)	A non-positive response or comment
FP (Feedback Positive)	A positive response or comment
IND (Indirect Instruction)	Implicit or polite instruction
OTH (Other)	Meaningless Words
QWH (Question What/How)	A who/what/when/where/why/how question
QYN (Questions Yes/No)	Questions asking for a yes/no answer
ST (Statement)	A declaration or remark.
UC (Uncertainty)	Dialogue that indicates uncertainty
Creativity Stages	**Definitions**
Clarify	Identify the goal, gather data to understand the goal, and formulate challenges
Idea	Generate ideas to solve challenges
Develop	Evaluate, strengthen, and select solutions for best "fit."
Implement	Support implementation of the selected solution(s).
RPP Roles	**Definitions**
Driver	Programmer writing code
Navigator	Programmer reviewing code
Discussing	Pairs planning together next steps

III. Lab Study

Supervised ML algorithms must be trained on a corpus of domain-specific labeled data to make accurate predictions. However, no such corpus exists for RPP, so our first step was to collect a dataset by conducting, transcribing, and labeling user sessions.

A. Participants

Participants were selected among university students with basic object-oriented programming experience. To avoid potential gender bias within the data, we recruited 18 students, nine of which self-identified as men and nine self-identified as women in their background questionnaires based on the definition of gender identity from [67], [68]. Therefore, we only focused on these two genders.

B. Study Design

The 18 participants were divided into pairs, resulting in nine sessions (3 man-man, 3 woman-woman, 3 man-woman). We distributed participants into same- and mixed-gender pairs because research has shown differences in communication styles between them [23]. The two participants of each pair were placed in separate rooms and collaborated using TeamViewer [69] to mimic an RPP environment. TeamViewer allowed participants to share a desktop and communicate through video, voice, and text.

In each session, participants were introduced and given time to practice pair programming and the think-aloud method [70], [71]. Participants were then asked to implement the game, Tic-Tac-Toe using the object-oriented programming language, Java. In Tic-Tac-Toe, two players, X and O,

take turns marking spaces in a 3x3 grid. We selected Java for two reasons: 1) it is the most popular programming language among professionals and students [72] and 2) it is used in introductory courses offered at our university. Sessions were recorded using Morae [73], a screen capture tool.

C. Analysis and Labeling

Recordings of each session were transcribed. Utterances were split by pauses in speech, rather than grammatically, to emulate the segmentation style of voice recognition technology commonly seen in applications such as Google Assistant [74] and Siri [75]. A total of 3,436 utterances were collected, and each was manually labeled with corresponding DAs, creativity stages, and RPP roles (refer Table I). When labeling RPP roles, we relied on keyboard activity from video captured of the participants. Two researchers independently labeled 20% of the transcripts and reached agreement on 75.7% of DAs, 73.7% of creativity stages, and 81.0% of RPP roles by calculating inter-rater reliability using the Jaccard measure [76]. The remaining data was split between the two researchers and independently labeled. The labeled dataset of our RPP sessions is available online [77]. On average, the transcribing and labeling (gender, timestamps, DAs, RPP roles, creativity stages) of an individual session took ~14 hours of consistent effort (~126 hours to label all 9 studies).

IV. Predicting DAs, Creativity Stages, and RPP Roles

A. Classification Algorithm

To evaluate the effectiveness of ML within the domain of RPP, we trained a supervised, multi-class text classification algorithm for each RQ (source code can be found at [78]). Multi-class classification assigns a single label to a sample of data from a set of many labels. For our ML algorithm, we used Support Vector Machines (SVMs) [79] for a variety of reasons: 1) as noted in [80], text data is well suited for SVMs, as the vector representation of text is often high-dimensional and sparse; 2) shallow machine learning algorithms such as SVMs require substantially less training data for patterns to emerge than deep learning algorithms such as neural networks [81]; 3) prior research on categorizing developer questions [42] has shown the viability of SVMs in a similar domain; and 4) researchers [42], [82], [83] recommend prioritizing a simple approach (SVMs) before moving to more advanced techniques (deep learning) when taking preliminary steps in a research area. Research on DA classification in phone conversations [43] found success using Hidden Markov Models (HMMs) in conversational dialogue, but after exploring both algorithms, we found that SVMs consistently outperformed HMMs.

B. Features

Our classifier's features derive from four sources: 1) current dialogue, 2) contextual dialogue, 3) shallow features, and 4) gender.

1) Dialogue: We vectorized both current and contextual utterances using TF-IDF (term frequency-inverse document frequency). Due to the high linear separability of TF-IDF features, we used a linear SVM kernel. Any single word, bi-gram, or tri-gram was included as a feature if it appeared at a frequency over 0.01% of all utterances, filtering out the less frequent ones to eliminate unnecessary noise. We preprocessed all utterances using stop word removal, lemmization, and stemming. We then combined similar words (e.g. uh/uhh, ok/okay) and merged all numbers (i.e. six, 9, forty-five) into the symbol "#."

2) Contextual: Pair programming dialogues can be sparse, and the meaning of dialogue is often context-dependent. Therefore, we explored different approaches to incorporate contextual features to accompany individual utterances.

- Approach #1 used the previous utterance, regardless of the speaker, as a contextual feature.
- Approach #2 used the other participant's previous dialogue.
- Approach #3 used the speaker's own previous dialogue.
- Approach #4 was a combination of #2 and #3.

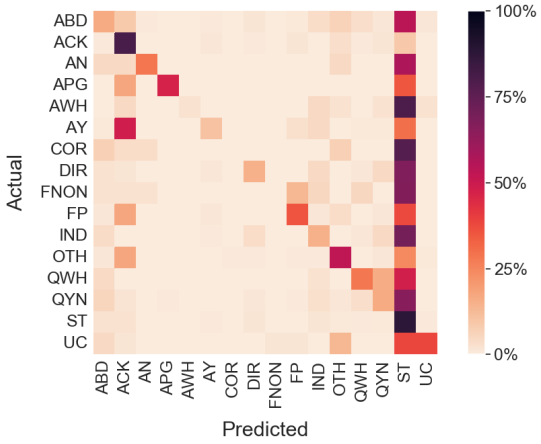
Approach #1 was useful when a small amount of "recent context" was needed. Approaches #2, #3, and #4 were helpful when "distant context" (i.e., utterances from the distant past) was necessary. In our analysis, we included a Control Approach that contained no contextual features to test the consistency of our results.

3) Shallow: To accompany dialogue features, we incorporated three shallow features identified in research on conversational text classification [84]–[86]: the position of the participant in the conversation, the number of words in an utterance, and the time between the previous and current utterance. However, several of the shallow features proposed by this research were not used, as they required knowledge of the future (e.g., calculating entropy over the entire conversation, the time until the next utterance, etc.) and therefore, cannot be applied to real-time applications, such as facilitator agents, where predictions can only be based on past and present information.

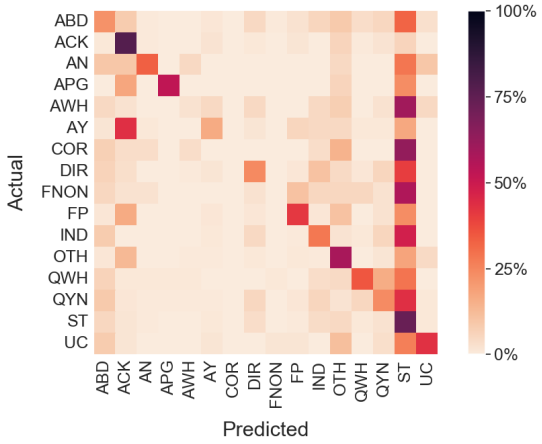
4) Gender: Given recent findings of gender-bias in AI [87], it was imperative to include gender as a feature, not only to avoid bias, but also to increase overall accuracy. Hence, we labeled each utterance with the gender of both participants. [23].

C. Evaluation Methodology

We used a 9-fold cross validation design, where each fold corresponded to 1 of the 9 RPP sessions. That is, in each validation run, 1 session was set aside as the testing set, while the remaining 8 sessions were used for training. Evaluation metrics were calculated 9 times (each session was used as the testing set once) and averaged. We chose this over randomized train-test splitting to avoid mixing sessions between training/testing data, as this would have disrupted our contextual features.



(a) Confusion Matrix without Weights.



(b) Confusion Matrix with Balanced Class Weights.

Fig. 1. The confusion matrix for predicting DAs in our RPP sessions. (a) The accuracy of ST (statements) is high: however, almost all other labels are incorrectly predicted as ST. (b) The accuracy of ST is lower, but other labels are more accurately predicted.

D. Evaluation Metrics

To evaluate the performance of our classifier in each RQ, we reported the overall accuracy score, as well as accuracy measurements for each label. We chose raw accuracy measurements over precision and recall because when designing facilitator agents, there is no reason to treat false-positive and false-negative errors differently. Therefore, accuracy is the preferred evaluation metric. We used chance as a baseline for our accuracy measurements. Chance is the accuracy attained by randomly picking labels (1/n were n is the number of unique labels).

We used confusion matrices to allow for a more detailed visualization of each classifiers' performance. Confusion matrices illustrate the "confusion," or the false-positives and false-negatives that occur between labels. We were interested in identifying what labels were misinterpreted as other labels. The rows within the matrix correspond to each label and the columns correspond to how the classifier interprets those labels. Correct predictions occur on the diagonal where the predicted and actual labels meet.

V. Prediction Results

RQ1: Classifying Dialogue Acts

In RQ1, we investigated whether we could accurately detect dialogue acts (DAs) within RPP sessions as a means to understand the intent of the speaker. Our classifier was able to classify DAs with an accuracy of 57.91%. Chance did not provide a meaningful baseline for our accuracy measurements in RQ1 because ST (statement) occurred frequently (1280/3436 = 37.25%, refer Table II).

A. Statements Dominate

To further understand the performance of our classifier, we calculated the accuracy of classifying each DA (refer Table II, column II) and generated the accompanying confusion matrix (refer Figure 1 (a)). The confusion matrix shows that our classifier disproportionately predicted ST: the most common label. To prevent this, we balanced the SVM's class weights so that labels' weights were inversely proportionate to their frequency, coercing the less common labels to be predicted more often. Using balanced class weights [79], the classifier was able to predict DAs with a 54.91% accuracy: 3.00% lower than without them (refer Table II). Although this decreased the accuracy of ST (-14.7%), it increased the accuracy of the less common labels, such as IND (+13.3%), DIR (+9.4%), QYN (+7.4%), QWH (+7.3%). Nonetheless, the least frequent labels (AWH, COR, and FNON) were still mistaken for ST almost every time. This is because ST occurred the most frequently (37.25% of all utterances), which allowed the classifier to identify more patterns associated with the ST label than with any other label. Since these patterns were not unique to ST, the classifier could not distinguish between ST and the least frequent labels.

Misinterpretation also occurred between two similar labels: AY and ACK. Utterances of both labels were typically variations of "yes." However, since AY occurred less frequently, it was often mistaken for ACK. This did not occur in reverse due to the discrepancy of the occurrence rates.

B. Contextual Features

To gain a better understanding of how the classifier learned from dialogue, we analyzed the effectiveness of encoding contextual information with and without balanced class weights (refer Table III).

Without balanced class weights, the highest accuracy was achieved without context, yet the improvement was minimal (only +1.5% from Approach #4). Without class weights, the classifier primarily focused on predicting the most frequent labels: ST and ACK.

However, when using balanced class weights, all contextual feature approaches lead to a drastic increase in accuracy, with Approach #1 performing the best (+6.77% the from Control Approach). In this case, "recent context" and "distant

TABLE II
ACCURACY SCORES FOR DAS WITHOUT AND WITH BALANCED CLASS WEIGHTS.

DAs	Normal Accuracy	Balanced Class Weight Accuracy	Occurrences (# Utterances)
ABD	16.6%	22.4%	205
ACK	80.8%	78.1%	621
AN	28.6%	33.3%	21
APG	47.1%	52.9%	17
AWH	2.5%	2.5%	40
AY	10.7%	16.5%	103
COR	0.0%	0.0%	27
DIR	15.1%	24.5%	159
FNON	0.0%	0.0%	37
FP	35.7%	41.7%	84
IND	15.1%	28.4%	218
OTH	52.9%	58.4%	238
QWH	28.2%	35.5%	110
QYN	16.7%	24.1%	216
ST	88.0%	73.3%	1280
UC	38.3%	43.3%	60
Avg / Total	57.9%	54.9%	3436

context" both became more important when predicting the less common labels (e.g., question/answer style dialogues).

Since Approach #1 was the most accurate, we analyzed results from our classifier using it and balanced class weights for the remainder of RQ1.

C. In-Depth Feature Importance Analysis

For SVM classifiers, using a linear kernel is especially useful for model analysis—compared to more sophisticated kernels like polynomial or radial basis function—since the coefficients of each feature correspond to their importance: how definitive they are to a particular label. Analyzing the most important features provided insight into how the classifier distinguished labels. In the case of DA classification, Figure 2 shows the feature importance for each label.

1) Descriptive Dialogue Features: The important features of some DAs made sense logically. For example, the top two features of QWH were entirely expected: "what" and "how." On the other hand, the least frequent labels such as FNON contained generic words that were not definitive to the label: "way," "get," and "okay." For these poor performing labels, the classifier did not have enough training data to identify adequate patterns.

2) Variable Dependency on Context: Of the top 10 most important features for each label, 75 of the total 180 (41.67%) were contextual (green cells in Figure 2). Reliance on context varied greatly by label: some DAs, such as ST and IND, contained no contextual features in their top 10 features. Others contained many; particularly DAs that relied on context by their very nature (i.e., answer dialogues AY, AN, AWH). This further suggested that balancing class weights and including contextual features was necessary for the classifier to learn non-ST labels well.

3) Domain Specific Features: One challenge of our research was the classifier's dependency on domain-specific words. To measure the extent of this dependency, we analyzed what percentage of the top 10 features were

	1	2	3	4	5	6	7	8	9	10
ABD	we would	each	board full	current	huh	we can	else	we go	let	loop
ACK	uh huh	ok	yes	okay	we would	alright	right	else	they	make
AN	no	uhh	how	not	when	else	uhh	could	know	current
APG	sorry	vertical	me	yes yes	method	know	oh	we should	sure	we need
AWH	what	row	board	else	how	# #	current player mark	you want	write	see
AY	we should	yes we	would	guess	we can	sure	yes	current player	good	you want
COR	no	actually	us	first	ok uh	we would	want	oh	not	number
DIR	let us	we should	we would	think we	get	change	let	yes we	thing	return
FNON	way	get	okay	let	not	we should	up	but	write	we could
FP	work	good	us	we go	when	think we	look	huh	make	actually
IND	we could	we should	add	need	write	could	maybe	set	start	think we
OTH	uh	put	think we	us	return	we could	column	ppau	what	player
QWH	what	how	mark	let us see	same	place	us see	case	you	put
QYN	current player	you	right	you want	something	we need	try	full	guess we	tie
ST	same	# # #	game	number	current	go	because	here	see	place mark
UC	know	sure	huh	we can	not	you want	same	ok uh	guess we	on

Fig. 2. The top 10 most important features when classifying DAs using contextual Approach #1. Feature types include the current utterance (gray), previous utterance (green), and ppau: time between previous and current turn (red).

domain-specific. Since the domain of our lab study was the game Tic-Tac-Toe, domain-specific words included "row," "column," "board," and "game."

Of the top 10 features for each label, 21 out of 180 (11.67%) were domain-specific words. However, they disproportionately came from the least frequent labels which means the dependency was even lower.

RQ2: Classifying Creativity Stages

We are interested in the detection of creativity stages within RPP as this knowledge would enable a facilitator agent to adjust its behavior depending on the creativity stage. For example, if an insufficient number of ideas are being generated with a RPP session, the agent could ask questions that prompt divergent thinking. Our classifier was able to predict the creativity stages of RPP dialogue with a 35.04% accuracy score, only 10.04% greater than chance (25%).

Our confusion matrix (refer Figure 3) shows that the `Clarify` and `Implement` stages were the most accurately predicted (40.4% and 51.4% respectively) as compared to the `Idea` (11.3%) and `Develop` (24.9%) stages. While this is due in-part to the lower occurrence rate (refer Figure 3), it illuminates the subjective nature of classifying creativity, since these stages often overlap. This also indicates the difficulty of classifying abstract concepts such as creativity using bag-of-words-based techniques because they heavily rely on keywords.

A. Contextual Features

Since creativity stages typically spanned multiple utterances, contextual features proved useful. All contextual

TABLE III
COMPARES THE ACCURACY OF DIFFERENT CONTEXTUAL APPROACHES FOR DAS, CREATIVITY STAGES, AND RPP ROLES.

Contextual Feature Approaches	DAs		Creativity	RPP Roles	
	Normal Accuracy	Balanced Class Weight Accuracy	Accuracy	Accuracy	Without Outlier
None (Control Approach)	**57.91%**	48.14%	32.35%	45.12%	**45.10%**
Approach #1 (Previous Dialogue Irresp. Of Participant)	56.69%	**54.91%**	**35.04%**	43.39%	42.21%
Approach #2 (Speaker's Previous Dialogue)	56.46%	53.91%	34.15%	44.39%	42.39%
Approach #3 (Other Participant's Previous Dialogue)	57.28%	54.50%	32.33%	44.16%	43.35%
Approach #4 (Both Participant's Previous Dialogue)	56.41%	54.80%	34.78%	**45.92%**	44.66%

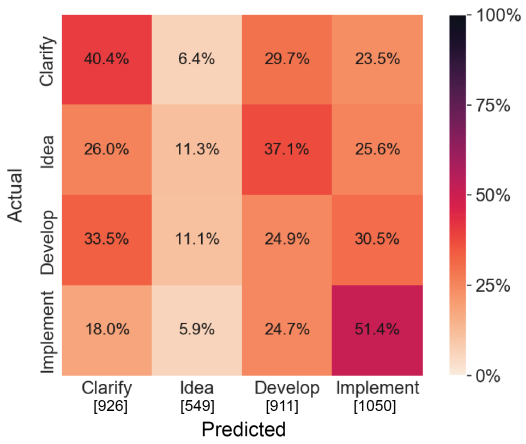


Fig. 3. The confusion matrix for predicting the creativity stages when using contextual Approach While `Clarify` and `Implement` were moderately predictable, `Idea` and `Develop` were not. The occurrence rate of each stage is shown in [] brackets.

	1	2	3	4	5	6	7	8	9	10
Clarify	run	run	look	thing	when	current player mark	game	write	mark	see
Idea	we could	we could	each	current player mark	maybe	we need	we need	huh	first	case
Develop	current player	we check	current player	same	same	player mark	you want	uh huh	# # #	out
Implement	equal	# #	current player mark	equal	we go	need	we go	need	else	actually

Fig. 4. The top 10 most important features when classifying creativity stages when using contextual Approach #1. Feature types include the current utterance (gray) and the previous utterance (green).

approaches (excluding #3) lead to an increase in accuracy (refer Table III).

Approach #1 outperformed Approach #4 (which encapsulated twice the contextual information) suggesting that the classification of creativity relied more on "recent context" rather than "distant context." This reliance results from the variable length of individual creativity stages, which either lasted across many utterances or changed after each one. When the creativity stage changed rapidly, "recent context" was preferred. This was also the reason why Approach #2 outperformed Approach #3: when the speaker said multiple utterances in a row (which was often the case), their previous dialogue was more recent than that of the other participant's. We used contextual Approach #1 for the remainder of RQ2 as it was the most accurate.

B. In-Depth Feature Importance Analysis

1) Few Descriptive Dialogue Features: Analysis of the feature importance for creativity classification revealed there were few words that directly linked to each creativity stage (refer Figure 4). For example, "run," the top feature for `Clarify`, related to an action taken within a development environment: running the code. In `Idea`, the speaker often used phrases like "we could" and "we need" when presenting new ideas. `Develop` contained "we check," which was a call to action on the code. `Implement` contained words uttered out-loud when writing code: "equal," "# #," and "else." While the existence of descriptive keywords shows that there were some low-level indicators for creativity detection, the classifier's poor performance suggests a dependency on high-level knowledge.

2) Reliance on Context: To measure the degree to which the classifier relies on context, we calculated the percentage of the top 40 features that were contextual; 73 out of 160 (45.63%) were from the previous dialogue. We considered a greater number of important features in RQ2 than in RQ1, since creativity stages had more important features to analyze compared to the least frequent DAs that had few (<15 important features).

3) Domain Specific Features: Of the 40 most important features from each label, 30 of the total 160 (18.75%) were domain-specific words. Relative to dialogue labeling, the classification of creativity stages was more domain-dependent (+7.08%).

RQ3: Classifying RPP roles

In RQ3, we investigated whether RPP roles can be automatically classified using dialogue alone. We rely on dialogue (rather than requiring other input) to keep the agent non-intrusive. Our classifier was able to predict RPP roles with an accuracy of 45.92%, 12.59% higher than chance (33.3%).

A. Weak Navigator Role

In Figure 5, the navigator role was shown to be both the least frequent (934/3436 = 27.18%) and least accurately predicted role (30.6% accuracy). This is in contrast to the accuracy of both other roles: around 51%.

B. Contextual Features

As seen in Table III, the best approach to encoding contextual information was to add the previous dialogue from both

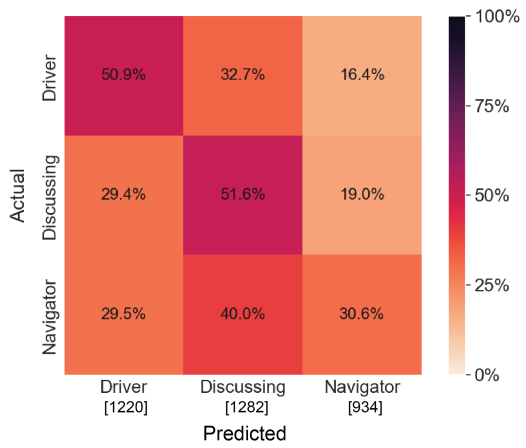


Fig. 5. The confusion matrix for predicting pair programming roles when using contextual Approach #4. `Navigator` was predicted less often than it was mistaken for `Discussing`. The occurrence rate of each stage is shown in [] brackets.

	1	2	3	4	5	6	7	8	9	10
Driver	yes we	current player mark	maybe	let	mark	good	Yes yes	ok uh	work	equal
Discussing	# # #	get	current	current player	when	we need	guess we	when	place mark	game
Navigator	we could	current player mark	current player mark	let	equal	first	good	know	full	win

Fig. 6. The 10 most important features for each label when classifying each pair programming role when using contextual Approach #4. Feature types include the current utterance (gray), speaker's own previous utterance (blue), and other participant's previous utterance (purple). "# # #" represents a sequence of three numbers.

participants to the original set of features (Approach #4). Our Control Approach (no contextual features) outperformed Approaches #1 - #3 and was only 0.80% less accurate than Approach #4.

Session 6 was an outlier because one participant hogged the `Driver` role throughout the session. Excluding this session causes all contextual approaches to perform slightly worse than the Control Approach (refer Table III). This highlights the importance of capturing the breadth of pair programming behavior when gathering training data.

We used contextual Approach #4 for the remainder of RQ3 as it was the most accurate when using all nine sessions.

C. In-Depth Feature Importance Analysis

1) Lack of Descriptive Dialogue Features: Analysis of the feature importance for RPP role classification revealed that no individual features stood out that were definitively unique to each category (refer Figure 6). This lack of descriptive keywords may be the cause of poor performance because bag-of-words-based techniques rely heavily on them.

2) Reliance on Context: One notable trend was the dependency on contextual information. Of the top 40 most important features for each label, 85 of the total 120 (70.83%) were contextual features. However, accuracy measurements do not support this claim, as there was only a +0.80% boost in performance with contextual features and a -0.44% drop without the outlier study (refer Table III).

3) Domain Specific Features: Of the 40 most important features for each label, 31 of the total 120 (25.83%) were domain-specific words.

VI. Discussions

A. Insights

1) The Importance of Contextual Features: In all RQs, the addition of contextual features often led to an increase in our classifier's accuracy. The best contextual approach for each RQ depended on how quickly the labels changed. Labels changed the most in RQ1, changed moderately in RQ2, and changed the least in RQ3. Likewise, "recent context" was preferred in RQ2, while "distant context" was preferred in RQ3. However, in RQ1, both "recent" and "distant" context proved equally effective. Furthermore, it was necessary to capture the turn-taking nature of conversation in RQ1 to predict response labels: AN, AY, and AWH. Contextual Approach #3 (the other participant's previous dialogue) encapsulated this dynamic and was more accurate than Approach #2 (the speaker's own previous dialogue).

2) The Correlation between Creativity Stages and RPP Roles: In RQ2 and RQ3, only two labels were predicted accurately (`Clarify` and `Implement` for RQ2; `Driver` and `Discussing` for RQ3). The two labels in each RQ may correlate to a clustering of high-level and low-level knowledge because pairs were `Discussing` high-level topics together in the `Clarify` stage, and the `Driver` explored code-related, low-level topics in the `Implement` stage.

3) No Effect of Gender: To identify whether the same approach to text classification can be used for both genders, we trained models separately by gender—i.e., training a classifier on men's data to test against women's data and vice-versa. However, we did not find any significant discrepancies between the accuracies of these gender-trained classifiers. We conjecture that the difference between genders was minimal because software/code-related dialogue is more standardized than normal dialogue. However in practice, it is unrealistic to collect gender-balanced data, especially in computer science [88], [89], creating a challenge for alleviating gender bias in future facilitator agents.

B. Challenges

1) Detecting RPP Roles with Dialogue: Our classifier relied solely on dialogue to make its predictions. However, past research [66] indicated that dialogue does not differ greatly between roles and that participants of both roles converse at the same level of abstraction. We also found it difficult to categorize RPP roles solely using dialogue, so instead, we relied on video footage of the participants to determine who was typing, since role exchange was not always verbal.

2) Dependence on Domain-Specific Words: A classifier trained across multiple domains cannot rely on domain-specific knowledge, and instead, must identify domain-independent patterns, creating a threat to external validity.

Our sessions share common verbiage unique to the game of Tic-Tac-Toe such as "column," "row," "player," and "mark." The extent to which a classifier relies on these domain-specific words to make its predictions determines its transferability: how much information from one domain can be used to predict in another. In our study, the rate of domain-specific words within the top features were: DAs 11.67%, creativity stages 18.75%, and RPP roles 25.83%. Hence, while we could train a classifier to detect DAs and creativity stages across different domains, it would be more difficult to do so for RPP roles.

3) Nuances in Dialogue: Intent classification is especially difficult in RPP conversations, as sentences were often not well thought-out, inadequately structured, and contained repeated words or phrases (e.g., woman participant WP9 said, *"For that part this is part this one and..."*; WP11 said, *"the other thing the only like other way is maybe like have like..."*; and man participant MP18 said, *"if all of them, if all of them are, yeah if all of them are..."*). This also may have contributed to the underperformance of HMMs (DAs: 39.84%, Creativity: 34.10%, and RPP Roles: 39.04%.). HMMs model linear transitions between discrete states and have proven effective in structured turn-taking phone conversations [43], but were less so in the sporadic conversational style of RPP. Furthermore, in current state-of-the-art speech recognition technology and research on automated transcription (e.g., [90]–[95]), voice recognition errors vary largely between individuals due to different speech patterns, accents, and volumes [96], increasing the challenge of identifying intent within RPP conversations. However, these challenges are not the focus of this paper.

4) The Need for a Large Corpus of Data for Sophisticated ML algorithms: Sophisticated ML algorithms (e.g., deep neural networks) are capable of achieving higher accuracy, but require a large corpus of training data before they eclipse the accuracy of shallow algorithms (e.g., SVMs). Deep-learning algorithms (e.g., RNN and CNN) are being used for DA classification of large corpus of switch board data [97]–[102]. However, our dataset consisted of only 3,436 utterances, so we could not take advantage of such algorithms, creating a threat to internal validity. We defer the exploration of deep learning algorithms to future research.

C. Implications

1) Domain and Language Specific Features: To increase the accuracy of classifiers used for software engineering applications, information regarding the domain (e.g., Tic-Tac-Toe, aviation management software, hospital management systems), programming language (e.g., Java, C, C++, SQL), and language type (e.g., object-oriented, declarative, machine and assembly languages) should be included as a feature to allow classifiers to interpret text features differently depending on the domain.

2) Voice Intonation and Facial Features: Our study utilized only dialogue features and did not consider other types of input such as voice intonation or facial expressions to aid the detection of emotions such as boredom, excitement, or confusion, as well as the power dynamics between pairs. This is especially important in analyzing and eliminating gender bias in RPP, since researchers found differences in communication styles (e.g., women use more non-verbal cues than men [23]). Therefore, our classifier's accuracy could be increased by including features that can detect emotion.

3) Keyboard Input as a Feature: Since we relied on video footage of participants' keyboards to identify RPP roles, the addition of keyboard input as a feature has the potential to improve the accuracy of RPP role classification. Additionally, the correlation between RPP roles and creativity stages suggests keyboard input might also improve the accuracy of creativity stage classification. Either RPP roles or direct keyboard input could be used as features for creativity stage classification.

4) Supporting Creativity in Education: The ability to accurately detect programmers' creativity stages could be used to facilitate learning in interactive education platforms (e.g., Codecademy [103]) and intelligent tutoring systems (e.g., [28], [104]). A facilitator agent could promote effective creative strategies by identifying which creativity stages novice programmers should focus on. Our creativity stage classifier could detect `Clarify` and `Implement` more accurately than `Idea` and `Develop`. Therefore, for educational platforms, creative facilitator agents could be designed to support two stages, but more research is required to increase the accuracy of classifying the `Idea` and `Develop` stages.

VII. Related Work

Within the discipline of software development, Wood et al. [42] envisioned the creation of virtual assistants like Siri, Cortana, and Google Assistant to aid programmers. They conducted a Wizard of Oz study where developers asked an assistant "Wizard" various questions. They then annotated speech acts—"spoken or written actions meant to accomplish a task [105]"—and later developed a classifier to detect them. Our research differs, as we envision a RPP facilitator agent which can classify remote pair programmers' conversations for DAs, creativity stages, and RPP roles.

VIII. Conclusions

Our study is the first to contribute to literature on RPP facilitator agents. We make the following contributions: (1) Making our labeled dataset of RPP conversations available for reproducibility by researchers and practitioners. (2) Providing insights on the feature selection, algorithms, and challenges when developing an ML classifier for detecting three facets of RPP dialogue: dialogue acts, creativity stages, and RPP roles. (3) Laying the foundation for the development of a facilitator agent. With this foundational work, we aim to empower programmers separated geographically.

References

[1] L. Williams and R. Kessler, *Pair Programming Illuminated*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[2] L. A. Williams and R. R. Kessler, "All I Really Need to Know About Pair Programming I Learned in Kindergarten," *Commun. ACM*, vol. 43, no. 5, pp. 108–114, May 2000.

[3] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Effects of Pair-programming on Performance in an Introductory Programming Course," in *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '02. ACM, 2002, pp. 38–42.

[4] K. Falkner, N. J. Falkner, and R. Vivian, "Collaborative Learning and Anxiety: A Phenomenographic Study of Collaborative Learning Activities," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. ACM, 2013, pp. 227–232.

[5] L. Porter and B. Simon, "Retaining Nearly One-third More Majors with a Trio of Instructional Best Practices in CS1," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. ACM, 2013, pp. 165–170.

[6] M. Celepkolu and K. E. Boyer, "Thematic analysis of students' reflections on pair programming in cs1," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. ACM, 2018, pp. 771–776.

[7] J. T. Nosek, "The Case for Collaborative Programming," *Commun. ACM*, vol. 41, no. 3, pp. 105–108, Mar. 1998.

[8] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "Pair Programming Improves Student Retention, Confidence, and Program Quality," *Commun. ACM*, vol. 49, no. 8, pp. 90–95, Aug. 2006.

[9] N. Nagappan, L. Williams, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik, "Improving the CS1 Experience with Pair Programming," in *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '03. ACM, 2003, pp. 359–362.

[10] T. Dybå, E. Arisholm, D. Sjøberg, J. Hannay, and F. Shull, "Are Two Heads Better than One? On the Effectiveness of Pair Programming," *Software, IEEE*, vol. 24, pp. 12 – 15, 12 2007.

[11] L. Cao, K. Mohan, P. Xu, and B. Ramesh, "How extreme does Extreme Programming Have to Be? Adapting XP Practices to Large-scale Projects," 02 2004, pp. 10 pp.–.

[12] L. L. Werner, B. Hanks, and C. McDowell, "Pair-programming Helps Female Computer Science Students," *J. Educ. Resour. Comput.*, vol. 4, no. 1, Mar. 2004.

[13] E. Allen, R. Cartwright, and C. Reis, "Production Programming in the Classroom," *SIGCSE Bull.*, vol. 35, no. 1, pp. 89–93, Jan. 2003.

[14] D. W. Palmieri, "Knowledge management through pair programming," 2002.

[15] Z. Li and E. Kraemer, "Social Effects of Pair Programming Mitigate Impact of Bounded Rationality," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '14. ACM, 2014, pp. 385–390.

[16] C. M. Lewis and N. Shah, "How Equity and Inequity Can Emerge in Pair Programming," in *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ser. ICER '15. ACM, 2015, pp. 41–50.

[17] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair-Programming," *Software, IEEE*, vol. 17, pp. 19 – 25, 08 2000.

[18] J. Chong and T. Hurlbutt, "The Social Dynamics of Pair Programming," in *Proceedings of the 29th International Conference on Software Engineering*, ser. ICSE '07. IEEE Computer Society, 2007, pp. 354–363.

[19] E. Arisholm, H. Gallis, T. Dybå, and D. Sjøberg, "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise," *Software Engineering, IEEE Transactions on*, vol. 33, pp. 65–86, 03 2007.

[20] L. Prechelt, U. Stärk, and S. Salinger, "7 Types of Cooperation Episodes in Side-by-Side Programming," 01 2009.

[21] E. Murphy-Hill and G. Murphy, "Peer interaction effectively, yet infrequently, enables programmers to discover new tools," 01 2011, pp. 405–414.

[22] D. Jones and S. Fleming, "What use is a backseat driver? A qualitative investigation of pair programming," 09 2013, pp. 103–110.

[23] S. Kaur Kuttal, K. Gerstner, and A. Bejarano, "Remote Pair Programming in Online CS Education: Investigating through a Gender Lens," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct 2019, pp. 75–85.

[24] A. Bandura, *Social foundations of thought and action : a social cognitive theory*. Prentice-Hall, 1986.

[25] N. Katira, L. Williams, L. Williams, E. Wiebe, C. Miller, S. Balik, and E. Gehringer, "On Understanding Compatibility of Student Pair Programmers," *SIGCSE Bull.*, vol. 36, no. 1, pp. 7–11, Mar. 2004.

[26] A. Begel and N. Nagappan, "Pair programming: What's in it For Me?" in *ESEM '08: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, October 2008, pp. 120–128.

[27] L. Williams, D. Mccrickard, L. Layman, and K. Hussein, "Eleven Guidelines for Implementing Pair Programming in the Classroom," 09 2008, pp. 445–452.

[28] M. Alavi, "Computer-Mediated Collaborative Learning: An Empirical Evaluation," *MIS Quarterly*, vol. 18, no. 2, pp. 159–174, 1994.

[29] K. M. Ying, L. G. Pezzullo, M. Ahmed, K. Crompton, J. Blanchard, and K. E. Boyer, "In Their Own Words: Gender Differences in Student Perceptions of Pair Programming," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '19. Association for Computing Machinery, 2019, p. 1053–1059.

[30] J. Tsan, J. Vandenberg, Z. Zakaria, J. Wiggins, A. Webber, A. Bradbury, C. Lynch, E. Wiebe, and K. Boyer, "A Comparison of Two Pair Programming Configurations for Upper Elementary Students," 03 2020.

[31] Z. Zakaria, D. Boulden, J. Vandenberg, J. Tsan, C. Lynch, E. Wiebe, and K. Boyer, "Collaborative Talk Across Two Pair-Programming Configurations," 06 2019.

[32] G. DeSanctis and R. B. Gallupe, "A Foundation for the Study of Group Decision Support Systems," *Management Science*, vol. 33, no. 5, pp. 589–609, May 1987.

[33] M. S. Poole, M. Homes, and G. DeSanctis, "Conflict Management and Group Decision Support Systems," in *Proceedings of the 1988 ACM conference on Computer-supported cooperative work - CSCW '88*. ACM Press, 1988.

[34] M. S. Poole, M. Holmes, R. Watson, and G. DeSanctis, "Group Decision Support Systems and Group Communication: A Comparison of Decision Making in Computer-Supported and Nonsupported Groups," *Communication Research*, vol. 20, no. 2, pp. 176–213, 1993.

[35] R. T. Watson, G. DeSanctis, and M. S. Poole, "Using a GDSS to Facilitate Group Consensus: Some Intended and Unintended Consequences," *MIS Quarterly*, vol. 12, no. 3, pp. 463–478, 1988.

[36] V. Sambamurthy and W. W. Chin, "The Effects of Group Attitudes Toward Alternative GDSS Designs on the Decision-making Performance of Computer-Supported Groups*," *Decision Sciences*, vol. 25, no. 2, pp. 215–241, 1994.

[37] M. Limayem, P. Banerjee, and L. Ma, "Impact of GDSS: Opening the Black Box," *Decision Support Systems*, vol. 42, pp. 945–957, 11 2006.

[38] T. X. Bui, *Co-oP: A Group Decision Support System for Cooperative Multiple Criteria Group Decision Making*. Springer Berlin Heidelberg, 1987.

[39] M. Aiken, M. Vanjami, and J. Krosp, "Group Decision Support Systems," vol. 16, no. 3, pp. 38+, 2020/3/2/ 1995.

[40] D. Vogel and J. Nunamaker, "Group Decision Support System impact: Multi-Methodological Exploration," *Information Management*, vol. 18, no. 1, pp. 15 – 28, 1990.

[41] P. Gray, "Group decision support systems," *Decision Support Systems*, vol. 3, no. 3, pp. 233–242, Sep. 1987.

[42] A. Wood, P. Rodeghero, A. Armaly, and C. McMillan, "Detecting Speech Act Types in Developer Question/Answer Conversations During Bug Repair," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. ACM Press, 2018.

[43] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer, "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech," *Computational linguistics*, vol. 26, no. 3, pp. 339–373, 2000.

[44] A. Vail and K. Boyer, "Identifying Effective Moves in Tutoring: On the Refinement of Dialogue Act Annotation Schemes," 06 2014, pp. 199–209.

[45] M. Hijjawi, Z. Bandar, and K. Crockett, "User's utterance classification using machine learning for arabic conversational agents," in *2013 5th International Conference on Computer Science and Information Technology*, March 2013, pp. 223–232.

[46] L. Seung-Ik and C. Sung-Bae, *An Intelligent Agent With Structured Pattern Matching for a Virtual Representative*, September 2001, pp. 305–309.

[47] E. Ribeiro, R. Ribeiro, and D. M. d. Matos, "Reconhecimento de actos de diálogo hierárquicos e multi-etiqueta em dados em espanhol," *Linguamática*, vol. 11, no. 1, p. 17–40, Jul 2019. [Online]. Available: http://dx.doi.org/10.21814/lm.11.1.278

[48] T. Wagner and R. A. Compton, *Creating Innovators: The making of young people who will change the world*. Simon and Schuster, 2012.

[49] Y. Zhao, *World class learners: Educating creative and entrepreneurial students*. Corwin Press, 2012.

[50] B. Edelman and Inc. (2010) Creativity and Education: Why it Matters. [Online]. Available: http://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_Creativity_and_Education_Why_It_Matters_study.pdf

[51] T. Brown, *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*. HarperBusiness, 2009.

[52] Z. Liu and D. J. Schonwetter, "Teaching Creativity in Engineering," *International Journal of Engineering Education*, vol. 20, no. 5, pp. 801–808, 2004.

[53] J. Austin, J. Austin, J. Urmson, J. Urmson, and M. Sbisà, *How to Do Things with Words*, ser. A Harvard paperback. Harvard University Press, 1975.

[54] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[55] C. Ford, B. Fox, and S. Thompson, *The Language of Turn and Sequence*, ser. Oxford Studies in Sociolinguistics. Oxford University Press, 2002.

[56] I. Hutchby and R. Wooffitt, *Conversation Analysis*. Polity, 2008.

[57] T. Nishida, *Conversational Informatics: An Engineering Approach*, ser. Wiley Series in Agent Technology. Wiley, 2008.

[58] P. T. Have, *Doing Conversation Analysis*, ser. Introducing Qualitative Methods series. SAGE Publications, 2007.

[59] F. J. Rodríguez, K. M. Price, and K. E. Boyer, "Exploring the Pair Programming Process: Characteristics of Effective Collaboration," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '17. Association for Computing Machinery, 2017, p. 507–512.

[60] S. G. Isaksen and D. J. Treffinger, "Celebrating 50 years of Reflective Practice: Versions of Creative Problem Solving," *The Journal of Creative Behavior*, vol. 38, no. 2, pp. 75–101, Jun. 2004.

[61] A. Osborn, *Applied Imagination: Principles and Procedures of Creative Thinking*. Charles Scribner's Sons, 1957.

[62] M. Page-Jones, *The Practical Guide to Structured Systems Design*, ser. Prentice-Hall international editions. Prentice Hall, 1988.

[63] R. K. Kavitha and M. S. I. Ahmed, "Knowledge Sharing Through Pair Programming in Learning Environments: An empirical study," *Education and Information Technologies*, vol. 20, no. 2, pp. 319–333, Oct. 2013.

[64] R. E. Mayer, "The Psychology of How Novices Learn Computer Programming," *ACM Comput. Surv.*, vol. 13, no. 1, pp. 121–141, Mar. 1981.

[65] D. N. Perkins and F. Martin, "Fragile Knowledge and Neglected Strategies in Novice Programmers," in *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*. Ablex Publishing Corp., 1986, pp. 213–229.

[66] L. Williams, "Integrating pair programming into a software development process." IEEE Comput. Soc.

[67] J. Butler, "Revisiting Bodies and Pleasures," *Theory, Culture & Society*, vol. 16, no. 2, pp. 11–20, 1999.

[68] C. WEST and D. H. ZIMMERMAN, "Doing Gender," *Gender & Society*, vol. 1, no. 2, pp. 125–151, 1987.

[69] "TeamViewer," 2019. [Online]. Available: https://www.teamviewer.com/en-us/

[70] C. H. Lewis, "Using the "Thinking Aloud" Method in Cognitive Interface Design," IBM, RC 9265, 1982.

[71] C. B. Seaman, "Qualitative Methods in Empirical Studies of Software Engineering," in *IEEE Transactions on Software Engineering*, 1999, pp. 557–572.

[72] "Popular Programming Language," 2020. [Online]. Available: https://www.tiobe.com/tiobe-index/

[73] "Morae," 2019. [Online]. Available: http://www.techsmith.com/morae.asp

[74] "Google Assistant," 2020. [Online]. Available: https://assistant.google.com/

[75] "Siri," 2020. [Online]. Available: https://www.apple.com/siri/

[76] P. Jaccard, "Etude de la distribution florale dans une portion des Alpes et du Jura," *Bulletin de la Societe Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, Jan 1901.

[77] "RPPDataset," 2020. [Online]. Available: https://docs.google.com/spreadsheets/d/1PFrX_ppNlsAKNmtoRkmBDuXl-kvMGKglBenHYZs9U0I

[78] "MLCode," 2020. [Online]. Available: https://github.com/grubtub19/PairBuddy-Classifier

[79] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information science and statistics. Springer (India) Private Limited.

[80] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *Proceedings of the 10th European Conference on Machine Learning*, ser. ECML'98. Springer-Verlag, 1998, p. 137–142.

[81] K. Pasupa and W. Sunhem, "A Comparison Between Shallow and Deep Architecture Classifiers on Small Dataset," in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Oct 2016, pp. 1–6.

[82] A. Armaly, J. Klaczynski, and C. McMillan, "A Case Study of Automated Feature Location Techniques for Industrial Cost Estimation," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Oct 2016, pp. 553–562.

[83] B. D. Cruz, B. Jayaraman, A. Dwarakanath, and C. McMillan, "Detecting Vague Words Phrases in Requirements Documents in a Multilingual Environment," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sep. 2017, pp. 233–242.

[84] G. Murray and G. Carenini, "Summarizing Spoken and Written Conversations," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Association for Computational Linguistics, 2008, p. 773–782.

[85] S. Rastkar, G. C. Murphy, and G. Murray, "Automatic Summarization of Bug Reports," *IEEE Trans. Softw. Eng.*, vol. 40, no. 4, p. 366–380, Apr. 2014.

[86] P. Rodeghero, S. Jiang, A. Armaly, and C. McMillan, "Detecting User Story Information in Developer-Client Conversations to Generate Extractive Summaries," in *Proceedings of the 39th International Conference on Software Engineering*, ser. ICSE '17. IEEE Press, 2017, p. 49–59.

[87] T. Sun, A. Gaut, S. Tang, Y. Huang, M. ElSherief, J. Zhao, D. Mirza, E. M. Belding, K. Chang, and W. Y. Wang, "Mitigating Gender Bias in Natural Language Processing: Literature Review," *CoRR*, vol. abs/1906.08976, 2019.

[88] "Gender Disparity," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Gender_disparity_in_computing

[89] "nsf women statistic."

[90] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, J. Chen, J. Chen, Z. Chen, M. Chrzanowski, A. Coates, G. Diamos, K. Ding, N. Du, E. Elsen, J. Engel, W. Fang, L. Fan, C. Fougner, L. Gao, C. Gong, A. Hannun, T. Han, L. Johannes, B. Jiang, C. Ju, B. Jun, P. LeGresley, L. Lin, J. Liu, Y. Liu, W. Li, X. Li, D. Ma, S. Narang, A. Ng, S. Ozair, Y. Peng, R. Prenger, S. Qian, Z. Quan, J. Raiman, V. Rao, S. Satheesh, D. Seetapun, S. Sengupta, K. Srinet, A. Sriram, H. Tang, L. Tang, C. Wang, J. Wang, K. Wang, Y. Wang, Z. Wang, Z. Wang, S. Wu, L. Wei, B. Xiao, W. Xie, Y. Xie, D. Yogatama, B. Yuan, J. Zhan, and Z. Zhu, "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 173–182. [Online]. Available: http://proceedings.mlr.press/v48/amodei16.html

[91] D. Yu and L. Deng, *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[92] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[93] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition,"

in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[94] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[95] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[96] M. Kubis, Z. Vetulani, M. Wypych, and T. Ziętkiewicz, "Open Challenge for Correcting Errors of Speech Recognition Systems," 2020.

[97] Z. Chen, R. Yang, Z. Zhao, D. Cai, and X. He, "Dialogue act recognition via crf-attentive structured network," *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, 2018.

[98] C. Bothe, C. Weber, S. Magg, and S. Wermter, "A context-based approach for dialogue act recognition using simple recurrent neural networks," *arXiv preprint arXiv:1805.06280*, 2018.

[99] A. Ahmadvand, J. I. Choi, and E. Agichtein, "Contextual dialogue act classification for open-domain conversational agents," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1273–1276. [Online]. Available: https://doi.org/10.1145/3331184.3331375

[100] A. Matsushima, N. Oka, C. Fukada, and K. Tanaka, "Understanding dialogue acts by bayesian inference and reinforcement learning," in *Proceedings of the 7th International Conference on Human-Agent Interaction*, ser. HAI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 262–264. [Online]. Available: https://doi.org/10.1145/3349537.3352786

[101] V. Raheja and J. Tetreault, "Dialogue act classification with context-aware self-attention," *arXiv preprint arXiv:1904.02594*, 2019.

[102] Q. Han, Y. Meng, F. Wu, and J. Li, "Non-autoregressive neural dialogue generation," *arXiv preprint arXiv:2002.04250*, 2020.

[103] "Codecademy," 2020. [Online]. Available: https://www.codecademy.com/

[104] R. W. Chabay and J. H. Larkin, *Computer assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches*. Routledge, 2020.

[105] L. Abbeduto, "Linguistic Communication and Speech Acts. Kent Bach amp; Robert M. Harnish. Cambridge: M.I.T. Press, 1979, Pp. xvii 327." *Applied Psycholinguistics*, vol. 4, no. 4, p. 397–407, 1983.