Navigating NLU Challenges in Pair Programming Agents: A Study on Data Size, Gender, Language, & Domain Effects

Jacob Hart[1], Jake AuBuchon[1], Shandler A. Mason[2] (✉), and Sandeep Kaur Kuttal[2,1]

[1] University of Tulsa, Tulsa OK 74104, USA
{jch389,jsa6790}@utulsa.edu
[2] North Carolina State University, Raleigh NC 27606, USA
{samason4,skuttal}@ncsu.edu

Abstract. Recent strides in Natural Language Understanding (NLU) for pair programming conversational agents underscore the importance of expanding datasets and constructing models applicable across genders, languages, and domains. The difficulty arises from the resource-intensive nature of gathering data through lab studies. Our study explores the potential use of vast amounts of pre-existing data for the training of conversational agents. We introduced software engineering-specific slot labels through an open-coding process by identifying key words and phrases. Our slot labels were integrated with a dataset of developer-developer (DD) and developer-agent (DA) utterances, annotated with software engineering-specific intent labels, from pair programming conversations. We employed the transformer-based language model, joint-BERT, to explore the required training size and gender-related impacts on intent and slot accuracy. To gauge the model's generalizability, we analyzed 5 pair programming video conversations sourced from YouTube. These conversations were fully labeled for software engineering-specific intent and slot, allowing us to investigate language and domain effects on the model's performance. Our findings reveal that 5 conversations, without a strict gender balance, can be used to train a pair programming NLU. Our study paves the way for expanding datasets used in the training of conversational agents.

Keywords: NLU · Conversational Agents · Pair Programming.

1 Introduction

Conversational agents have transformed our daily lives by enhancing accessibility [60,37], fostering emotional connections (e.g., Cleverbot [56], Xiaoice [77], Mitsuku [57]), facilitating customer service [1], and supporting in routine tasks (e.g., Alexa [63], Apple's Siri [64], Google Assistant [65]). Past studies have explored the benefits of designing a pair programming conversational agent [33,30,32] and investigated the feasibility of training NLU using developer-developer (DD) [52]

and developer-agent (DA) [51] conversations. These studies released a conversational dataset, comprising 7,879 utterances marked with software engineering-specific intent labels, capturing developers collaborating on the development of a Java Tic-Tac-Toe game. However, effective language model training demands a more extensive benchmark dataset derived from developer conversations, similar to ATIS [27], SNIPS [14], and MultiWoZ [11] spanning over multiple domains.

Furthermore, for language models to exhibit generalizability across genders, programming languages, and domains, there is a need for diverse datasets. However, the collection of conversational data, from developers, poses both a costly and time-intensive challenge for both researchers and practitioners. We build upon our earlier research [51], in which we established that (DD) conversations can be utilized to train conversational agents, to extend our findings by delving into the impact of dataset size, gender, language, and domain on model accuracy.

Our study marks an initial step in establishing software engineering-specific slot labels, which is a crucial component for identifying key words or phrases during the implementation of a comprehensive NLU system. Building upon our existing dataset of (DD) and (DA) conversations, we employed an open-coding method to formulate a slot labeling framework tailored for pair programming conversations. We applied our slot labeling framework to annotate pair programming conversations from prior research.

To examine the impact of dataset size and gender, leveraging existing data from [51], we formulated two research questions:

- **RQ1: How much data is required to train a pair programming NLU?** The size of the training dataset is crucial for models to achieve consistent accuracy. We employed jointBERT, to facilitate simultaneous slot tagging and intent classification. We aimed to explore the optimal training size for datasets.
- **RQ2: How does gender effect the performance of a pair programming NLU?** Prior research has highlighted gender differences while pair programming [30]; thus, we investigated whether maintaining a strict gender balance is necessary for the training and testing of models.

To examine the viability of leveraging existing developer-developer (DD) conversations from online video hosting platforms, we formulated one research question:

- **RQ3: How should developer-developer pair programming conversations, from online videos, be incorporated into the training dataset of a pair programming NLU?** We searched YouTube for recent videos featuring developers engaged in pair programming, with the goal of exploring the potential for expanding pair programming conversational datasets through existing videos.

2 Background & Related Works

2.1 Bots for Developers

Bots help developers perform several tasks such as assisting with debugging [2], identifying syntax errors [48,42], automating repetitive tasks [3], mining repositories [4], locating Stack Overflow posts [73], and providing code feedback [70]. Developers use bots to perform load testing [43] and suggest patches based on failed test cases [62]. Bots can facilitate the on-boarding process for new project members [55], locate and utilize REST APIs [19], design micro-service based architectures [36], and assist in agile retrospectives [40].

2.2 Design & Feasibility of Pair Programming Agents

In our earlier work [44,53], we established design guidelines for a pair programming agent by implementing a lab study and a Wizard of Oz (WoZ) study, where participants engaged with an agent whose behaviors was covertly orchestrated by a researcher [15]. These guidelines encompass practices such as clearly acknowledging suggestions, offering feedback in a positive tone, expressing uncertainty through both verbal and non-verbal cues, and apologizing for mistakes [53]. We discovered that an agent effectively facilitated knowledge transfer, leading to developers displaying increased trust and humility without a significant change in code quality, productivity, or self-efficacy. Based on our findings, we further investigated the feasibility of a pair programming [33] and facilitator agent [52]. We observed comparable performance among transformer-based language models, with BERT having a slight advantage. Our findings indicated that (DD) conversations were effective for training the intent classifier of a pair programming NLU with optimal performance being achieved by training with (DD) conversations and fine-tuning with (DA) conversations.

2.3 Dialogue Datasets

Researchers have labeled and annotated data in the realm of software design and development, with a focus on intent classification. For instance, the Ubuntu dialogue corpus [39] comprises unstructured human-human chats extracted from chat logs, where researchers aimed to do next utterance classification without considering slots. In another study, Viviani et al. [66] delved into design and decision-related dialogues extracted from pull requests and online discussions, focusing on intent perspective. Ebert et al. [18] and Pascarella et al. [45] identified the types of questions posed by software engineers during code reviews. Wood et al. [72,71] utilized open-coding to devise an intent labeling scheme for debugging conversations. Previously, we employed an open-coding process to develop a hierarchical approach to intent labeling, resulting in 26 unique intent labels tailored for pair programming conversations [51].

Beyond the realm of software development, various non-software engineering datasets such as SWITCHBOARD [23], ATIS [27,61], SNIPS [14] are employed to train models for intent and slot classification.

2.4 Slot Labeling

A fully implemented NLU must have the capability of slot tagging, which is the process of identifying important words in a sentence. Slots enable the agent to pinpoint essential parts of an utterance [31,75]. Slots originated in the GUS architecture [29], grounded in the concept of frames. In GUS, the agent identifies the user's intent and utilizes the ongoing conversation to complete all necessary slot values before making a query. In the modern dialogue state architecture, slots serve to delimit the conversation space which empowers the agent to ask clarifying questions, offer suggestions, or respond based on the evolving conversation rather than adhering to a predetermined template [29]. Slots enable models to understand the typical locations of key terms within a specific intent. Slot tagging is framed as a sequence-to-sequence modeling problem, with the input sequence being the utterance and the output sequence consisting of associated slot tags [22]. Slots offer the advantage of modeling occurrences of key terms within a specific intent, and defining the potential values of the key terms. This precision aids the model in determining whether a slot has been accurately identified.

2.5 Transfer Learning on Intermediate Tasks

Transfer learning involves training on a source task followed by fine-tuning on a target task [76,10]. An additional training task, referred to as an intermediate task, can be incorporated within the transfer learning process. Phang et al. [46] introduced STILTs (Supplementary Training on Intermediate Labeled-data Tasks) which entails training a model initially on an unlabeled dataset, then on an intermediate labeled dataset, and ultimately fine-tuning the model on the target task. They found that STILTs can enhance the performance of large language models such as BERT [16], ELMo [10], and GPT [49].

Researchers, including Sap et al. [54] and Clark et al. [13], have explored STILTs by using intermediate tasks to enhance model performance. Wang et al. [67] conducted a comprehensive study with ELMo and BERT, revealing the complexities of multitask learning and the impact of dataset size on target task performance. Pruksachatkun et al. [47] experimented with BERT and RoBERTa, emphasizing the ambiguity of training on intermediate tasks.

3 Methodology

3.1 Developer-Developer and Developer-Agent Conversations

Robe et al. [51] released a dataset comprising 7,879 utterances from 9 (DD) and 14 (DA) pair programming conversations collected through a remote environment lab study [52] and a (WoZ) study [51]. In both studies, they conducted 40 minute sessions where participants engaged in pair programming to implement a Tic-Tac-Toe game in Java. Participants utilized the think-aloud method [35] and adhered to test-driven development principles [7]. All participants were

equipped with template code, including the game board, a sample test case, and user stories for the programming task.

The (DD) dataset was curated through recorded conversations in a remote pair programming lab study with 18 participants [52]. These participants, exclusively computer science students, were strategically paired by their self-identified gender, resulting in 9 gender-balanced pairs: 3 men-men, 3 men-women, and 3 women-women.

The (DA) conversation dataset were captured using the (WoZ) methodology. Yang et al. [74] affirm (WoZ) as the singularly, viable method for collecting conversations with an agent prior to its actual implementation because it facilitates iterative designs and the observation of user behaviors. (WoZ) serves as a foundational tool for training machine learning algorithms [15] and has found widespread application in natural language interfaces, including conversational agents [9,72,74]. The (WoZ) study involved 14 participants, 6 computer science students (3 men, 3 women), and 8 professionals (4 men, 4 women) [51]. The motivation and implementation of the agent, in the (WoZ) study, was inspired by Robe et. al [53].

All (DD) and (DA) transcripts were labeled with software engineering-specific intent labels and developed using a hierarchical, open-coding process [8]. The absence of software engineering-specific slot labels serves as motivation for our work.

3.2 Slot Labeling

We developed our slot labels employing an open coding process, following the practices of other software engineering researchers [52,72]. Table 1 is the list of our slot labels (boolean, feedback, filename, filename_method, current_error, driver, inequality, keyword, line_number, location, name_of_user, number, objective, phase_of_sdl, text_add, text_remove, user_story, variable), along with their descriptions.

To establish the initial set of slot labels, two researchers analyzed three (DA) studies using the open-coding method to generate potential relevant slots. The two researchers discussed the potential slot labels, consolidated similar labels, and independently labeled a (DA) conversation. In a series of iterations, the researchers discussed and reached agreement on new slot labels as they emerged during their analysis. To ensure inter-rater reliability, two researchers labeled 20% of the (DD) and (DA) data, achieving a Cohen's Kappa of 0.7192, indicating substantial agreement [41]. The remaining data was labeled by one researcher. We used Inside-Outside-Beginning (IOB) tagging in the slot labeling process [50]. Each study took approximately 2 hours for manual slot labeling, accumulating to 46 total hours.

3.3 Model Design

To perform simultaneous slot labeling and intent classification, we employed jointBERT, an extension of BERT developed by Chen et al. [12]. BERT is a

Table 1: Our slot labels and descriptions. Domain dependent slots are annotated with 'D' and language dependent slots are annotated with 'L.' Both language and domain dependent slots are annotated with 'L, D.' The corresponding numbers show the frequency of label occurrences in the (DA), (DD), and (YT) dataset.

Label Name	Description/Example	Dependence	(DA)	(DD)	(YT)
boolean	Used to help understand relations between objects and their function (e.g., true, false).		160	183	0
feedback	This allows the developer to agree or disagree with a suggestion, question, or clarification and continue with their thoughts (e.g., yes, no, good, bad).		265	544	75
filename	Name of the file being discussed but not necessarily the file currently open.	L, D	94	38	117
filename_method	Name of the method being discussed. The filename helps to differentiate polymorphic methods.	L, D	517	454	15
current_error	The reason why the program is not running as expected.	L, D	36	2	6
driver	Determines if the user or the agent is driving.		46	3	0
inequality	Used to help understand relations between objects and their function (e.g., >=, <=, >, <, =).		90	122	4
keyword	Used to help understand relations between objects and their function with language denoted keywords or reserved words.	L	344	413	202
line_number	Identifies the specific location in the code that is under discussion.		112	13	20
location	Refers to the documentation or an application (e.g., web browser, terminal window).	D	38	17	0
name_of_user	Used to identify and personalize communication with the user.		0	0	10
number	Used to help understand relations between objects and their function (e.g., numbers in assignment, comparison, or other functions within the code).		499	649	27
objective	The current next step. Multiple objective work towards a user story.	D	705	707	135
phase_of_sdl	High-level definition of the current phase in the software engineering development life cycle (e.g., plan, analyze, design, implement, test, maintain).		54	18	0
text_add	Allows the user to write code verbally using speech.		0	0	12
text_remove	Allows the user to remove code verbally using speech.		0	0	2
user_story	Used to identify the current user story which is a high-level goal that contains multiple objectives.		323	90	85
variable	Used to capture user-defined variables and its functionality.	L, D	244	450	37

bidirectional transformer-based model that was pre-trained on masked language modeling and next sentence prediction tasks using the BooksCorpus [78] and English Wikipedia [16]. Previously, BERT has been applied to software analysis [68], technology comparison tools via online discussions [69], and machine translation failure detection [25]. The uncased BERT model was expanded to jointBERT, which was evaluated on the SNIPS [14] and ATIS [27,61] datasets.

Implementation. To enhance the utility of jointBERT, our model was trained on the entire intent label, contrasting with our previous approach [51], where a model for each category of intent was employed in a pipeline manner. Training on the entire intent label simplifies the learning objective for jointBERT. Moreover, a comprehensive pair programming agent can leverage a multi-model approach to NLU with jointBERT providing full intent classification and specialized models for validation. Our implementation utilized HuggingFace's `Transformers`

Table 2: Our model's 5-fold cross validation accuracy for slot and intent, along with the intent F1 score for (DA), (DD), and (DD→DA).

	DA	**DD**	**DD→DA**
Intent	68.39	55.38	68.72
Intent F1	70.85	56.96	71.12
Slot	99.16	98.58	99.08

Python package, specifically the `bert-based-uncased` model. We utilized the pre-trained BERT tokenizer with an `encode_length` of 66.

We conducted a hyperparameter sweep, a process involving training models with all possible combinations within the search space [47]. We explored various learning rates (1e-3, 1e-4, 5e-5, 1e-5), epsilons (1e-6, 1e-7, 1e-8, 1e-9), and batch sizes (8, 16, 32, 64, 128) with both `Adam` and `AdamW` optimization algorithms. This results in 250 combinations, each using 5-fold cross-validation, resulting in 1,250 trained models. Our search space was motivated by Chen et al [12], who trained for (1, 5, 10, 20, 30, 40) epochs using an `Adam` optimizer with a learning rate of 5e-5 and a batch size of 128. They observed that jointBERT, with 1 epoch of training, outperformed other slot-predicting models such as LSTMs [26], Attention-Based BiDirection RNNs [38], and Slot-Gated [24]. Based on our hyperparameter sweep results, we trained our model using `Adam` with a learning rate of 5e-5, epsilon of 1e-9, and a batch size of 16, for 12 epochs.

Performance. Table 2 illustrates the slot, intent accuracy, and intent F1 score when the model was trained on (DA), (DD), and employed transfer-learning by training on (DD) then fine-tuning on (DA) (DD→DA). We reported all metrics as the average of the 5-fold cross-validation using new training and testing sets. We implemented the `KFold` method from SKLearn's `model_selection` library. We found performance variation between (DA) (Intent F1: 70.85%) and (DD) (Intent F1: 56.96%), which is similar to our previous hierarchical model [44]. We observed minor improvements in the (DA) dataset (Intent F1: 70.85%) when we used (DD→DA) (Intent F1: 71.12%), also similar to our previous work [44]. The slot accuracy for (DA) was 99.16% and (DD) was 98.58%; however, for (DD→DA) the slot accuracy slightly decreased to 99.08%. Still, the (DD→DA) slot accuracy (99.08%) was superior to the (DD→DA) intent accuracy (68.72%), which suggests that training on (DD) then fine-tuning on (DA) remains the best option to train a pair programming NLU.

4 Results

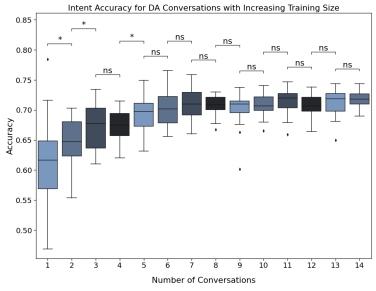
4.1 RQ1: How much data is required to train a pair programming NLU?

The size of the training data, used in machine learning models, plays a pivotal role in achieving consistent accuracy. Transformer-based language models, like

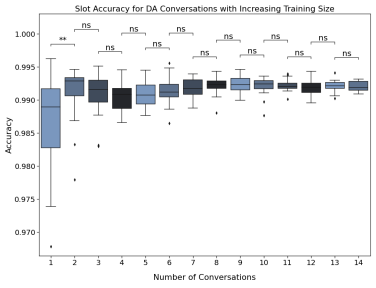
those pre-trained on extensive corpora such as BookCorpus [78] and Wikipedia, require less training data. Fine-tuning is essential for adapting these models to specific downstream tasks [10]. Our training dataset, consisting of 3,436 (DD) and 4,443 (DA) utterances, falls between the range of standard AI data benchmarks, including ATIS (5,871 utterances) [27,61] and SNIPS (16,000 queries) [14]. Thus, we explored the impact of data size on intent and slot accuracy.

Methodology. We used pair programming conversations in our training dataset because it serves as a robust metric for enhancing existing conversational data and tends to be more elaborate compared to other task-oriented conversational agents (e.g., Siri [64], Alexa [63], Google Assistant [65]). We incrementally trained our models on each (DD) or (DA) conversation, employing 5-fold cross-validation in randomized order for each iteration, resulting in 25 total models.

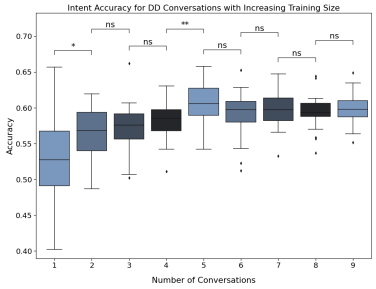
Fig. 1: Intent (1a, 1c) and slot accuracy (1b, 1d) of (DA) and (DD) conversations when increasing the training size. '*' represents p-value $\leq .05$; '**' represents p-value $\leq .01$; '***' represents p-value $\leq .001$; '****' represents p-value $\leq .0001$; 'ns' represents p-value $> .05$.



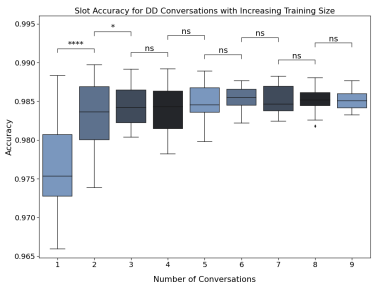
(a) Intent accuracy for (DA)



(b) Slot accuracy for (DA)



(c) Intent accuracy for (DD)



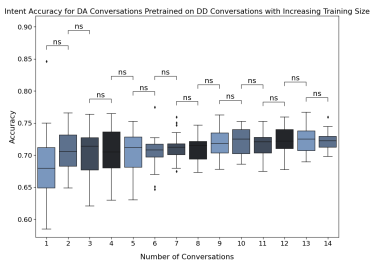
(d) Slot accuracy for (DD)

Findings. Fig. 1 illustrates the impact of dataset size on intent and slot accuracy. The box plots represent the intent (Fig. 1a, 1c) or slot (Fig. 1b, 1d) accuracy for our 25 models. The outliers, in Fig. 1, may be attributed to idealized test cases, where portions of the test dialogue included novel examples not encountered during training, or the inherent variability of conversation. We performed a paired t-test when (DA) or (DD) conversations were incorporated into the training dataset to assess differences in the model's performance. We represented statistically significant p-values with a '*' and non-significant p-values with 'ns'.

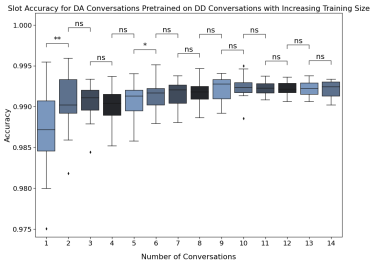
As represented in Fig. 1, both (DA) and (DD) show that a transformer-based language model requires a minimum of 5 pair conversational units. To verify the accuracy of our findings, we performed a one-way analysis of variance (ANOVA) to identify statistically significant differences across groups [58]. This analysis focused on the performance of models trained with 5 or more conversations, resulting in non-significant p-values of 0.31 for (DA) and 0.55 for (DD). In Fig. 1b and 1d, the slot performance, required 2 pair conversational units for (DA) and 3 for (DD) before performance leveled. We conducted ANOVA testing on the models' performance after training with 5 or more conversations, which was the minimum required for intent. We found non-significant differences for (DA) with a p-value of 0.18 and (DD) with a p-value of 0.84.

Fig. 2 represents the impact of dataset size on intent and slot accuracy of transfer learning, (DD→DA), models. We discovered that the intent accuracy of 1 (DA) conversation, in the (DD→DA) model, matches that achieved with 5-6 (DA) conversations (refer to Fig. 1a). Based on our results from the paired t-test, we found a non-significant change, for intent accuracy when adding one pair conversation to the training data set. We performed ANOVA testing, with 14 conversations, and found a significant difference with a p-value of 0.000014. For the slot accuracy, we found a significant difference from 4 to 5 conversations, based on the paired t-test. We used ANOVA testing on the remaining conversations which gave a significant p-value of 0.024.

Fig. 2: Intent (2a) and slot (2b) accuracy for (DD→DA).



(a) Intent accuracy for (DD→DA)



(b) Slot accuracy for (DD→DA)

Summary of RQ1. Based on our findings, 5 conversations encompassing 1,500-2,000 utterances total, are required to train a pair programming NLU, irrespective of transfer learning. This aligns with the findings of Huggins et al. [28], demonstrating that a high performance with BERT can be achieved with 25 examples per intent. We did not explore data augmentation techniques that possibly could reduce the total number of utterances needed to get a similar, stable performance.

4.2 RQ2: How does gender effect the performance of a pair programming NLU?

Collaboration is a critical component of software development. Efficient collaboration may be effected by developers gender, race, or geographical location [17,6]. We investigated how gender may effect the performance of a pair programming NLU because gender gaps exist in computing-related classrooms (18% women) and workplaces (10% women) [59]. Previous studies have investigated differences in problem-solving [5], communication [30], and leadership style [30] between men and women while pair programming. Furthermore, while numerous pair programming videos are accessible on the internet for training our NLU, the majority of these videos feature men-men pairs. To leverage these videos for our model training, we assessed the generalizability of a model primarily trained on data from men-men pairs to mitigate the risk of perpetuating gender bias.

Methodology. We separated the (DD) and (DA) datasets based on the self-identified gender of the speaking participant. We combined the data, by gender, for (DD), (DA), and (DD→DA) to assess the model's overall performance. We created 6 total datasets: M-DD (9 men utterances); M-DA (7 men utterances); M-DD→DA (16 men utterances); W-DD (9 women utterances); W-DA (7 women utterances); and W-DD→DA (16 women utterances). We trained and tested 12 models within the same conversational-context group (DA, DD, DD→DA). We reported accuracy, for same-gender comparisons (M-M, W-W), as the average of the 5-fold cross-validation. To analyze mixed-gender comparisons (M-W, W-M), we trained the model on the first gender dataset and tested on the second gender dataset in a one-to-one comparison.

Findings. Table 3 illustrates the 12 models intent and slot accuracy performance. Our analysis found gender bias across all conversational context groups (DA), (DD), (DD→DA).

Training and testing exclusively with men data resulted in better performance, for intent accuracy, irrespective of conversational context. For example, within the DD conversational context, the alternative training and testing methods (W-W, W-M, M-W) exhibited accuracy below 60%; whereas, the (M-M) accuracy was 86.62%. Moreover, when training on men data and testing on women data (M-W), we observed performance levels comparable to those

achieved when training and testing exclusively on women data (W-W). To illustrate, for the (DA) dataset, (M-W) intent accuracy was 69.09% and (W-W) intent accuracy was 70.79%. For the (DD) dataset, (M-W) intent accuracy was 59.54% and (W-W) intent accuracy was 58.44%.

Training and testing with exclusively women data (W-W) resulted in the best slot accuracy performance, for (DA) with 99.88% and (DD) with 98.91%. For (DD→DA), the best performance came from exclusively men data (M-M) with 98.95%. For (DA) the slot accuracy was comparable for (M-M) with 99.67%, (W-W) with 99.88%, and (M-W) with 99.67%.

Summary of RQ2. Our findings indicate that while men data can be utilized for training a pair programming NLU, the inclusion of women data enhances overall performance. These results are promising in light of the underrepresentation of women in computing classes, workplaces, and online videos.

Table 3: Intent and slot accuracy based on same- and mixed-gender training and testing datasets (W-W, W-M, M-W, M-M) and (DA), (DD), (DD→DA) conversational contexts.

		DA		**DD**		**DD→DA**	
Train	Test	Intent	Slot	Intent	Slot	Intent	Slot
W	W	70.79	99.88	58.44	98.91	64.48	98.69
W	M	67.16	99.35	53.78	98.15	61.87	98.04
M	W	69.09	99.67	59.54	98.75	61.93	98.50
M	M	88.77	99.67	86.62	98.75	87.20	98.95

4.3 RQ3: How should developer-developer pair programming conversations, from online videos, be incorporated into the training dataset of a pair programming NLU?

A pair programming agent is required to support diverse domains and languages. However, our collected data is centered around one language (Java) and a specific domain (a Tic-Tac-Toe game). Robe et al. [44] demonstrated that (DD) data is usable for intent detection, but (DA) data is needed for achieving higher accuracy. Further, our findings in RQ1 show that slot accuracy slightly decreases when using transfer-learning. Considering these challenges and findings, we selected 5 pair programming videos, from YouTube, to investigate the generalizability of the slot, intent labels and feasibility of using online videos as a training dataset.

Methodology. We found pair programming videos by using a private browsing window of Google.com to search the term 'pair programming'. Then, we used built-in filtering tools to find videos published since 2016 that were longer than

20 minutes. We used this filtering criteria because newer videos are likely to focus on current popular languages, domains and longer videos are more comparable to the existing data set with comprehensive pair programming sessions. We selected 3 types of candidate videos: (1) the same language but a different domain; (2) similar domain but different language; (3) different language and domain. The similar domain mimicked a Tic-Tac-Toe game by placing objects on a grid. We reviewed all candidate videos to confirm their language was English, ensure their availability, and verify the presence of two participants. The videos represent approximately 350 minutes of pair programming conversations and 4,822 utterances.

The video we used with the same language but from a different domain featured 2 open-source software developers working through pull-requests, error submissions, and merge requests for a Java library that functioned as a verification tool [20]. The pair interaction was formal, and they navigated through tasks systematically with the intent to record their interaction for posterity, potentially serving as a valuable resource for historians, researchers, or their own future reference.

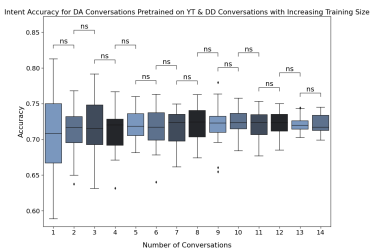
The first video we used from a similar domain but with a different language featured 2 coworkers, one man and one woman, who aimed to demonstrate the basics of creating a 2D interactive terrain for a game board. The pair worked in a casual and humorous manner to introduce variables and console inputs to their audience of beginner programmers. They used JavaScript and CSS in their demonstration. The second video we used was a virtual meeting between an online teacher and student. The pair debugged the student's tower defense game to create a working version. The interaction included frequent sarcasm, from the teacher, which appeared to negatively impact the students' responses. The target audience for this video was self-taught programmers. The pair used JavaScript and CSS for the project.

The first video we used from a different language and domain was of 2 friends collaborating to create a Facebook Messenger Bot that 'echoes' messages to users connected to the bot. The interaction was friendly with a lighthearted atmosphere. The second video we used was of 2 Kaggle (a competitive venue for data science collaboration) partners who live streamed their code development for a drug classification competition. They used Python and the `Pytorch` library to demonstrate the process of building a neural network, catering to an audience of Kagglers and self-taught programmers. The pair interaction consisted of formal help and role-switching request with minimal interruptions.

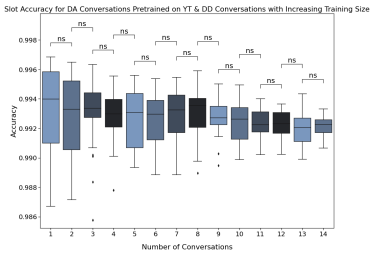
We transcribed all 5 videos using YouTube's auto-generated closed captioning. We manually adjusted phrasing errors while labeling. We conducted intent and slot labeling for each transcript. Each researcher labeled 20% of the data independently and reached a Cohen's Kappa of 0.810 for the intent labels and 0.725 for the slot labels, which is considered substantial agreement [41]. The remaining transcripts were divided between two researchers, one researcher labeled 3 videos for intent and 2 for slot, and the other research labeled 2 videos for intent and 3 for slot. The researchers dedicated 14 hours to annotate each

video for both intent and slot, totaling to 70 hours. Similar to RQ1, we repeated the 5-fold cross validation 5 times by randomizing the order of the studies.

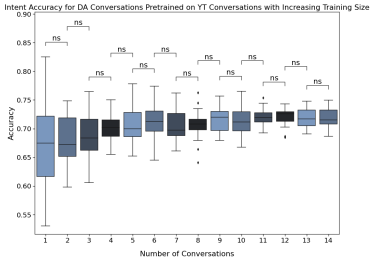
Fig. 3: Intent and slot accuracy for (YT→DA) and (YT→DD→DA) conversations when increasing the training size.



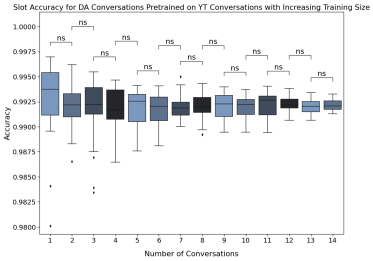
(a) Intent accuracy for (YT→DD→DA)



(b) Slot accuracy for (YT→DD→DA)



(c) Intent accuracy for (YT→DA)



(d) Slot accuracy for (YT→DA)

Findings. Previous research [44] and our current findings discovered that transfer learning (DD→DA), results in better model performance. Further, in RQ1, we found that 5 conversations, representing about 1,500-2,500 utterances, is sufficient to level-off jointBERT performance; however, that is representative of one language and domain. Therefore, we explored three ways of using transfer learning: (1) training on YouTube (YT) data, then fine-tuning on (DA), (YT→DA); (2) training on YouTube (YT) data, then training on (DD), then fine-tuning(DA), (YT→DD→DA); (3) training on YouTube (YT) data, then fine-tuning on (DA), (YT→DA).

Table 4 presents the intent and slot accuracy for (DA), (DD→DA), (YT→DA), and (YT→DD→DA). We found that by training on (YT) data, (YT→DD→DA) and (YT→DA), we achieved comparable, slightly improved performance for intent accuracy when compared to (DD→DA). For slot, the performance slightly increased with the inclusion of (DD) from 99.14% for (YT→DA) to 99.17% for (YT→DD→DA).

Figure 3 compares (YT→DA) and (YT→DD→DA) intent and slot accuracy when increasing the training size. We used ANOVA on all conversations and found a non-significant p-value for (YT→DD→DA) intent accuracy (p-value = 0.90), refer to Fig. 3a. We found a non-significant p-value for (YT→DD→DA) slot accuracy (p-value = 0.37), refer to Fig. 3b.

Table 4: The intent and slot accuracy of the model when trained on (DA), (DD→DA), (YT→DD→DA), and (YT→DA).

	DA	**DD→DA**	**YT→DD→DA**	**YT→DA**
Intent	68.39	68.72	68.77	68.79
Slot	99.16	99.08	99.17	99.14

Summary of RQ3. Our findings suggest that relying solely on (DD) and (DA) datasets is insufficient for training a pair programming NLU. Initially training with diverse data from (YT) and fine-tuning for a specific domain is the optimal approach to maximize performance. Thus, future endeavors should focus on expanding the collection of labeled video transcripts, ensuring the inclusion of new domains and languages overtime.

5 Limitations

A limitation of our study is the utilization of a linear model, in contrast to previous work that used a hierarchical modeling approach [51]. In a hierarchical modeling approach, the model is trained for each node of the intent hierarchy. Adopting this approach with jointBERT could complicate the learning objective or result in the prediction of slots without the full intent. Potential threats to validity may stem from our video selection process. The YouTube videos we selected may not encompass all diverse domains and languages, but our goal was to ensure that the videos closely resembled situations that a future agent might encounter. The sample size of five videos may be considered small, but the videos contains 4,822 utterances from 10 programmers across 4 domains and 3 languages. Furthermore, we present a set of software engineering specific slot labels which may be insufficient at representing all slots in pair-programming conversations. Labeling errors may have occurred in the manual labeling process, but this was mitigated by using an iterative open-coding process and data validation tools. The researchers who labeled the data self-identified as men, potentially introducing implicit bias into the labeling process. Furthermore, the size of the encoding length may have impacted slot accuracy. If the conversational context leads to shorter utterances on average, this would involve additional padding for each utterance, potentially making it easier for the model to predict the correct slot label.

6 Discussion

For RQ1, we investigated how much data was needed to train a pair programming NLU. Prior work by Sap et al. [54] found that increasing the training size of the intermediate task resulted in better performance on their downstream task; therefore, we focused on the creation of a dataset for the intermediate task. We found that training on 1,500-2,000 utterances, across 5 conversations, resulted in stable model performance. This highlights the importance of considering the number of utterances and conversations when training a pair programming NLU. For example, the two extremes are one conversation with 1,500-2,000 utterances and hundreds of short conversations with a total of 1,500-2,000 utterances. The first extreme, involving only one conversation, would make training on infrequent labels, such as greetings, difficult; whereas, the other extreme would fail to capture more nuanced labels, such as those relating to task control. Our findings have implications for training other agents to have longer form conversations.

A major advantage to expanding the dataset through pre-existing data is the reduction in cost; however, as the dataset expands, maintaining a gender balance will become increasingly difficult. Reinforcing bias within AI is recognized in facial recognition [34], bots [21], and underlying language models such as BERT [16]. This problem is confounded by the lack of women in computing classrooms and industry positions. For RQ2, we examined if the gender bias inherent in BERT affected our model and explored the optimal strategy for further expanding the dataset to enhance overall performance. Our findings show that datasets can deviate from a strict gender balance between men and women, but periodic checks remain essential to mitigating the risk of perpetuating gender bias.

For RQ3, we investigated two methods of integrating pair programming conversations from YouTube. Our first method involved training the model on (YT) and (DD) before fine-tuning with (DA); whereas, our second method solely trained on (YT) before fine-tuning with (DA). The (YT→DA) method, without (DD), required 1,500-2,000 (DA) utterances, whereas the (YT→DD→DA) method, including (DD), required 300-400 (DA) utterances to level off performance. While our first method achieved comparable performance, with less (DA) utterances, both methods require a similar number of total utterances. Our findings indicate that conversations from YouTube could serve as a starting point for exploring multiple languages and domains in the development of pair programming conversational datasets.

7 Conclusion

Our study determined the required training data size (RQ1), scrutinized potential gender bias (RQ2), and assessed the viability of leveraging online videos featuring developers engaged pair programming (RQ3), for the creation of a pair programming NLU.

Our research contributes to the broader Software Engineering and Human-Computer Interaction community with our software engineering-specific slot labeling scheme. We expanded the dataset, from earlier work, to include slot labels

for the original (DA) and (DD) conversations. Our study introduced 5 fully labeled pair programming conversations from an online video hosting platform. We explored various training methods to optimize performance, aiming to incorporate general conversational data. Our results has implications for minimizing the costs associated with conducting and transcribing lab studies, facilitating the expansion of pair programming conversational datasets, and training future NLUs for pair programming.

Acknowledgements. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0108 and National Science Foundation under award numbers IIS-2313890 and CCF-2006977. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the NSF and AFOSR.

References

1. Chatbot Statistics (2020), https://www.smallbizgenius.net/by-the-numbers/chatbot-statistics/#gref
2. Jetbrains (2021), https://www.jetbrains.com/
3. Visual studio (2021), https://visualstudio.microsoft.com/
4. Abdellatif, A., Shihab, E.: Msrbot: Using bots to answer questions from software repositories. Empirical Software Engineering **25**, 1834–1863 (2020)
5. Arisholm, E., Gallis, H., Dybå, T., Sjøberg, D.: Evaluating pair programming with respect to system complexity and programmer expertise. IEEE TSE **33**, 65–86 (03 2007)
6. Arnaoudova, V., Haiduc, S., Marcus, A., Antoniol, G.: The use of text retrieval and natural language processing in software engineering. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. vol. 2, pp. 949–950 (2015). https://doi.org/10.1109/ICSE.2015.301
7. Beck, K.: Test Driven Development: By Example. Addison-Wesley Longman Publishing Co., Inc. (2002)
8. Berg, B.L., Lune, H.: Qualitative Research Method for the Social Sciences. Pearson Education Limited (2017)
9. Bickmore, T., Cassell, J.: Relational agents: a model and implementation of building user trust. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 396–403 (2001)
10. Bowman, S.R., Pavlick, E., Grave, E., Van Durme, B., Wang, A., Hula, J., Xia, P., Pappagari, R., McCoy, R.T., Patel, R., et al.: Looking for elmo's friends: Sentence-level pretraining beyond language modeling (2018)
11. Budzianowski, P., Wen, T., Tseng, B., Casanueva, I., Ultes, S., Ramadan, O., Gasic, M.: Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. CoRR **abs/1810.00278** (2018), http://arxiv.org/abs/1810.00278
12. Chen, Q., Zhuo, Z., Wang, W.: BERT for joint intent classification and slot filling. CoRR **abs/1902.10909** (2019), http://arxiv.org/abs/1902.10909

13. Clark, C., Lee, K., Chang, M.W., Kwiatkowski, T., Collins, M., Toutanova, K.: BoolQ: Exploring the surprising difficulty of natural yes/no questions. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 2924–2936. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1300, https://aclanthology.org/N19-1300
14. Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., Primet, M., Dureau, J.: Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. CoRR **abs/1805.10190** (2018), http://arxiv.org/abs/1805.10190
15. Dahlbäck, N., Jönsson, A., Ahrenberg, L.: Wizard of oz studies: why and how. In: International conference on Intelligent user interfaces. pp. 193–200 (1993)
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
17. Dzvonyar, D., Alperowitz, L., Henze, D., Bruegge, B.: Team composition in software engineering project courses. p. 16–23. SEEM '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3194779.3194782, https://doi.org/10.1145/3194779.3194782
18. Ebert, F., Castor, F., Novielli, N., Serebrenik, A.: Communicative intention in code review questions. In: ICSME. pp. 519–523. IEEE (2018)
19. Ed-Douibi, H., et al.: Openapi bot: A chatbot to help you understand rest apis. In: Web Engineering. pp. 538–542. Springer International Publishing (2020)
20. Falco, L.: Approval Test Java Repository (2022), https://github.com/approvals/ApprovalTests.Java
21. Feine, J., Gnewuch, U., Morana, S., Maedche, A.: Gender bias in chatbot design. In: Chatbot Research and Design: Third International Workshop, CONVERSATIONS 2019, Amsterdam, The Netherlands, November 19–20, 2019, Revised Selected Papers 3. pp. 79–93. Springer (2020)
22. Gao, J., Galley, M., Li, L.: Neural approaches to conversational ai. Foundations and Trends® in Information Retrieval **13**(2-3), 127–298 (2019). https://doi.org/10.1561/1500000074, http://dx.doi.org/10.1561/1500000074
23. Godfrey, J.J., Holliman, E.C., McDaniel, J.: Switchboard: Telephone speech corpus for research and development. In: Acoustics, speech, and signal processing, ieee international conference on. vol. 1, pp. 517–520. IEEE Computer Society (1992)
24. Goo, C.W., Gao, G., Hsu, Y.K., Huo, C.L., Chen, T.C., Hsu, K.W., Chen, Y.N.: Slot-gated modeling for joint slot filling and intent prediction. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). pp. 753–757. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). https://doi.org/10.18653/v1/N18-2118, https://aclanthology.org/N18-2118
25. Gupta, S., He, P., Meister, C., Su, Z.: Machine Translation Testing via Pathological Invariance, p. 863–875 (2020)
26. Hakkani-Tür, D., Tür, G., Celikyilmaz, A., Chen, Y.N., Gao, J., Deng, L., Wang, Y.Y.: Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In: Interspeech. pp. 715–719 (2016)

27. Hemphill, C.T., Godfrey, J.J., Doddington, G.R.: The ATIS spoken language systems pilot corpus. In: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990 (1990), https://aclanthology.org/H90-1021
28. Huggins, M., Alghowinem, S., Jeong, S., Colon-Hernandez, P., Breazeal, C., Park, H.W.: Practical guidelines for intent recognition: Bert with minimal training data evaluated in real-world hri application. In: Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction. pp. 341–350 (2021)
29. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, USA, 3rd edn. (2023)
30. Kaur Kuttal, S., Gerstner, K., Bejarano, A.: Remote pair programming in online cs education: Investigating through a gender lens. In: VL/HCC. pp. 75–85 (2019)
31. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
32. Kuttal, S.K., Myers, J., Gurka, S., Magar, D., Piorkowski, D., Bellamy, R.: Towards designing conversational agents for pair programming: Accounting for creativity strategies and conversational styles. In: VL/HCC. pp. 1–11 (2020)
33. 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. In: CHI (2021)
34. Kyriakou, K., Kleanthous, S., Otterbacher, J., Papadopoulos, G.A.: Emotion-based stereotypes in image analysis services. In: Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization. pp. 252–259 (2020)
35. Lewis, C.: Using the "thinking-aloud" method in cognitive interface design. IBM T.J. Watson Research Center (1982)
36. Lin, C.T., Ma, S.P., Huang, Y.W.: MSABot: A Chatbot Framework for Assisting in the Development and Operation of Microservice-Based Systems, p. 36–40. ACM, New York, NY, USA (2020)
37. Lister, K., Coughlan, T., Iniesto, F., Freear, N., Devine, P.: Accessible conversational user interfaces: Considerations for design. In: International Web for All Conference. ACM (2020)
38. Liu, B., Lane, I.R.: Attention-based recurrent neural network models for joint intent detection and slot filling. CoRR **abs/1609.01454** (2016), http://arxiv.org/abs/1609.01454
39. Lowe, R., Pow, N., Serban, I., Pineau, J.: The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. arXiv preprint arXiv:1506.08909 (2015)
40. Matthies, C., Dobrigkeit, F., Hesse, G.: An additional set of (automated) eyes: Chatbots for agile retrospectives. In: BotSE. p. 34–37. BotSE '19, IEEE Press (2019)
41. McHugh, M.L.: Interrater reliability: the kappa statistic. Biochemia medica **22**(3), 276–282 (2012)
42. Memeti, S., Pllana, S.: Papa: A parallel programming assistant powered by ibm watson cognitive computing technology. Journal of Computational Science **26**, 275–284 (2018). https://doi.org/https://doi.org/10.1016/j.jocs.2018.01.001, https://www.sciencedirect.com/science/article/pii/S1877750317311493
43. Okanović, D., et al.: Can a chatbot support software engineers with load testing? approach and experiences. In: ICPE. p. 120–129. ACM, New York, NY, USA (2020)

44. P. Robe, S. K. Kuttal, J.A., Hart, J.: Pair programming conversations with agents vs. developers: Challenges & opportunities for se community. In: The ACM Joint European Software Engineering Conference, and Symposium on the Foundations of Software Engineering (2022)
45. Pascarella, L., Spadini, D., Palomba, F., Bruntink, M., Bacchelli, A.: Information needs in contemporary code review. Proc. ACM Hum.-Comput. Interact. (2018)
46. Phang, J., Févry, T., Bowman, S.R.: Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. CoRR **abs/1811.01088** (2018), http://arxiv.org/abs/1811.01088
47. Pruksachatkun, Y., Phang, J., Liu, H., Htut, P.M., Zhang, X., Pang, R.Y., Vania, C., Kann, K., Bowman, S.R.: Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? CoRR **abs/2005.00628** (2020), https://arxiv.org/abs/2005.00628
48. Queirós, R.A.P., Leal, J.P.: Petcha: a programming exercises teaching assistant. In: Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education. pp. 192–197 (2012)
49. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
50. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Natural language processing using very large corpora, pp. 157–176. Springer (1999)
51. Robe, P., AuBuchon, J., Kuttal, S.K., Hart, J.: Pair programming conversations with agents vs. developers:challenges opportunities for se community. In: FSE (2022)
52. Robe, P., Kaur Kuttal, S., Zhang, Y., Bellamy, R.: Can machine learning facilitate remote pair programming? challenges, insights & implications. In: VL/HCC. pp. 1–11 (2020)
53. Robe, P., Kuttal, S.K.: Designing PairBuddy – Conversational Agent for Pair Programming, vol. 29 (may 2022)
54. Sap, M., Rashkin, H., Chen, D., Le Bras, R., Choi, Y.: Social IQa: Commonsense reasoning about social interactions. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 4463–4473. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1454, https://aclanthology.org/D19-1454
55. Serrano Alves, L.P., Wiese, I.S., Chaves, A.P., Steinmacher, I.: How to find my task? chatbot to assist newcomers in choosing tasks in oss projects. In: Chatbot Research and Design. pp. 90–107. Springer International Publishing (2022)
56. Cleverbot, https://www.cleverbot.com/
57. Mitsuku, https://www.pandorabots.com/mitsuku/
58. St, L., Wold, S., et al.: Analysis of variance (anova). Chemometrics and intelligent laboratory systems **6**(4), 259–272 (1989)
59. Strachan, R., Peixoto, A., Emembolu, I., Restivo, M.T.: Women in engineering: Addressing the gender gap, exploring trust and our unconscious bias. In: IEEE Global Engineering Education Conference. pp. 2088–2093 (2018)
60. Torres, C., Franklin, W., Martins, L.: Accessibility in Chatbots: The State of the Art in Favor of Users with Visual Impairment, pp. 623–635 (2019)
61. Tur, G., Hakkani-Tür, D., Heck, L.: What is left to be understood in atis? In: 2010 IEEE Spoken Language Technology Workshop. pp. 19–24 (2010). https://doi.org/10.1109/SLT.2010.5700816

62. Urli, S., Yu, Z., Seinturier, L., Monperrus, M.: How to design a program repair bot? insights from the repairnator project. In: ICSE-SEIP. p. 95–104. ICSE-SEIP '18, ACM, New York, NY, USA (2018)
63. Amazon Alexa, https://developer.amazon.com/en-US/alexa
64. Apple Siri, https://www.apple.com/siri/
65. Google Assistant, https://assistant.google.com/
66. Viviani, G., Famelis, M., Xia, X., Janik-Jones, C., Murphy, G.: Locating latent design information in developer discussions: a study on pull requests. IEEE TSC pp. 1402–1413 (2019)
67. Wang, A., Hula, J., Xia, P., Pappagari, R., McCoy, R.T., Patel, R., Kim, N., Tenney, I., Huang, Y., Yu, K., Jin, S., Chen, B., Van Durme, B., Grave, E., Pavlick, E., Bowman, S.R.: Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 4465–4476. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1439, https://aclanthology.org/P19-1439
68. Wang, D., Dong, W., Li, S.: A Multi-Task Representation Learning Approach for Source Code, p. 1–2 (2020)
69. Wang, H., Chen, C., Xing, Z., Grundy, J.: DiffTech: A Tool for Differencing Similar Technologies from Question-and-Answer Discussions, p. 1576–1580 (2020)
70. Williams, A.C., Kaur, H., Iqbal, S., White, R.W., Teevan, J., Fourney, A.: Mercury: Empowering programmers' mobile work practices with microproductivity. In: UIST. p. 81–94 (2019)
71. Wood, A., Eberhart, Z., McMillan, C.: Dialogue Act Classification for Virtual Agents for Software Engineers during Debugging, p. 462–469. ACM (2020)
72. Wood, A., Rodeghero, P., Armaly, A., McMillan, C.: 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)
73. Xu, B., Xing, Z., Xia, X., Lo, D.: Answerbot: Automated generation of answer summary to developers' technical questions. In: ASE. pp. 706–716 (2017)
74. Yang, Q., Steinfeld, A., Rosé, C., Zimmerman, J.: Re-Examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design, p. 1–13. ACM (2020)
75. Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., Shi, Y.: Spoken language understanding using long short-term memory neural networks. In: 2014 IEEE Spoken Language Technology Workshop (SLT). pp. 189–194. IEEE (2014)
76. Zhang, J., Zhao, T., Yu, Z.: Multimodal hierarchical reinforcement learning policy for task-oriented visual dialog. In: Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue. pp. 140–150. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/W18-5015, https://aclanthology.org/W18-5015
77. Zhou, L., Gao, J., Li, D., Shum, H.Y.: The Design and Implementation of XiaoIce, an Empathetic Social Chatbot. Computational Linguistics **46**(1), 53–93 (03 2020). https://doi.org/10.1162/coli_a_00368, https://doi.org/10.1162/coli_a_00368
78. Zhu, Y., Kiros, R., Zemel, R.S., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S.: Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. CoRR **abs/1506.06724** (2015), http://arxiv.org/abs/1506.06724