Conversation Learner – A Machine Teaching Tool for Building Dialog Managers for Task-Oriented Dialog Systems

† **Swadheen Shukla***   † **Lars Liden***   † **Shahin Shayandeh***   ‡ **Eslam Kamal**
† **Jinchao Li**   † **Matt Mazzola**   † **Thomas Park**   † **Baolin Peng**   † **Jianfeng Gao**

† Microsoft Research AI
‡ Microsoft Corporation

{swads,laliden,shahins,eskam,jincli,mattm,thpar,bapeng,jfgao}@microsoft.com

Abstract

Traditionally, industry solutions for building a task-oriented dialog system have relied on helping dialog authors define rule-based dialog managers, represented as dialog flows. While dialog flows are intuitively interpretable and good for simple scenarios, they fall short of performance in terms of the flexibility needed to handle complex dialogs. On the other hand, purely machine-learned models can handle complex dialogs, but they are considered to be black boxes and require large amounts of training data. In this demonstration, we showcase *Conversation Learner*, a machine teaching tool for building dialog managers. It combines the best of both approaches by enabling dialog authors to create a dialog flow using familiar tools, converting the dialog flow into a parametric model (e.g., neural networks), and allowing dialog authors to improve the dialog manager (i.e., the parametric model) over time by leveraging user-system dialog logs as training data through a machine teaching interface.

1 Introduction

The proliferation of messaging applications and hardware devices with personal assistants has spurred the imagination of many in the technology industry to create task-oriented dialog systems that help users complete a wide range of tasks through natural language conversations. Tasks include customer support, IT helpdesk, information retrieval, appointment booking, etc. The wide variety of tasks has created the need for a flexible task-oriented dialog system development platform that can support many different use cases, while remaining simple for developers to use and maintain.

A task-oriented dialog system is typically built as a combination of three discrete systems, performing language understanding (for identifying user intent and extracting associated information), dialog management (for guiding users towards task completion), and language generation (for converting agent actions to natural-language system responses). The Dialog Manager (DM) contains two sub-systems: the Dialog State Tracker (DST) for keeping track of the current dialog state, and the Dialog Policy (DP) for determining the next action to be taken in a given dialog instance. The DP relies on the internal state provided by DST to select an action, which can be a response to the user, or some operation on the back-end database (DB). In this paper, we present a novel approach to building dialog managers (DMs).
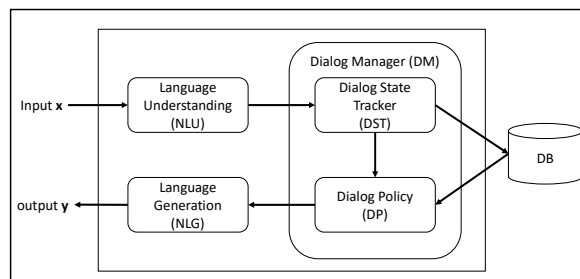

Figure 1: An architecture for a task-oriented dialog system.

In a typical industrial implementation of a task-oriented dialog system, the DM is expressed as a *dialog flow*, which is often a finite state machine, with nodes representing dialog activities (system actions) and edges representing conditions (dialog states that represent the previous user-system interactions). Since a dialog flow can be viewed as a set of *rules* that specify the flow between dialog states, it may also be called a *rule-based DM*.

There has been an increasing need for tools to help dialog authors[1] develop and maintain rule-

---

*Equal contribution.

[1]In this paper, "author" may refer to developers, business owners, or domain experts who define and maintain the conversational aspects of a task-oriented dialog system.

based DMs. These tools are often implemented as drag-and-drop WYSIWYG tools that allow users to specify and visualize all the details of the dialog flow. They often have deep integration with popular Integrated Development Environments (IDEs) as editing frontends. Examples of rule-based or partially rule-based DMs include Microsoft's Power Virtual Agents (PVA) [2] and Bot Framework (BF) Composer [3], Google's Dialog Flow [4], IBM's Watson Assistant [5], Facebook's Wit.ai [6] and Amazon's Lex [7]. It should be noted that most of these tools have some built-in machine-learned NLU capabilities, i.e. intent classification and entity detection, that can be leveraged to trigger different rule-based dialog flows, e.g. asking appropriate questions based on missing slots from the dialog state.

However, a rule-based DM suffers from two major problems. First, these systems can have difficulty handling complex dialogs. Second, updating a rule-based DM to handle unexpected user responses and off-track conversations is often difficult due to the rigid structure of the dialog flow, the long-tail (sparseness) of user-system dialogs, and the complexity in jumping to unrelated parts of the flow.

In end-to-end approaches proposed recently (Madotto et al., 2018; Lei et al., 2018), the DM is implemented as a neural network model that is trained directly on text transcripts of dialogs. Gao et al. (2019) presents a survey of recent approaches. One benefit provided by using a neural network model is that the network infers a latent representation of dialog state, eliminating the need for explicitly specifying dialog states. Neural-based DMs has been an area of active development for the research community as well as in industry; PyDial (Ultes et al., 2017), ParlAI (Miller et al., 2017), Plato (Papangelis et al., 2020), Rasa (Bocklisch et al., 2017), DeepPavlov (Burtsev et al., 2018), and ConvLab (Lee et al., 2019) are a few examples. However, these machine-learned neural DMs are often viewed as black boxes from which dialog authors have difficulty interpreting why individual use cases succeed or fail. Further, these approaches often lack a general mechanism for accepting task-

specific knowledge and constraints, thus requiring a large number of validated dialog transcripts for training. Collection and curation of this type of corpus is often infeasible.

This paper presents Conversation Learner, a machine *teaching* tool for building DMs, which combines the strengths of both rule-based and machine-learned approaches. Conversation Learner is based on Hybrid Code Networks (HCNs) (Williams et al., 2017) and the *machine teaching* discipline (Simard et al., 2017). Conversation Learner allows dialog authors to (1) import a dialog flow developed using popular dialog composers, (2) convert the dialog flow to an HCN-based DM, (3) continuously improve the HCN-based DM by reviewing user-system dialog logs and providing updates via a machine teaching UI, and (4) convert the (revised) HCN-based DM back into a dialog flow for further editing and verification.

Section 2 describes the architecture and main components of Conversation Learner. Section 3 demonstrates Conversation Learner features. Section 4 presents a case study of using Conversation Learner as the DM of a text-based customer support dialog system.

2 Conversation Learner

Development of any DM follows an iterative process of generation, testing, and revision. Conversation Learner follows a three-stage DM development process:

1. Dialog authors develop a rule-based DM (dialog flow) using a dialog composer.

2. The DM is imported into a HCN dialog system. Users (or human subjects recruited for system fine-tuning) interact with the system and generate user-system dialog logs.

3. Dialog authors revise the DM by selecting representative failed dialogs from the logs and teaching the system to complete these dialogs successfully. Run regression testing. Return to step 2.

This development process is illustrated in Figure 2 (Bottom). The overall architecture of Conversation Learner is shown in Figure 2 (Top). It consists of four components: (1) a DM converter that converts a dialog flow between rule-based and HCN-based DM representations; (2) an HCN-based DM engine; (3) a machine teaching module

---

[2] https://powervirtualagents.microsoft.com/
[3] https://github.com/microsoft/BotFramework-Composer
[4] https://dialogflow.com/
[5] https://www.ibm.com/watson/
[6] https://wit.ai/
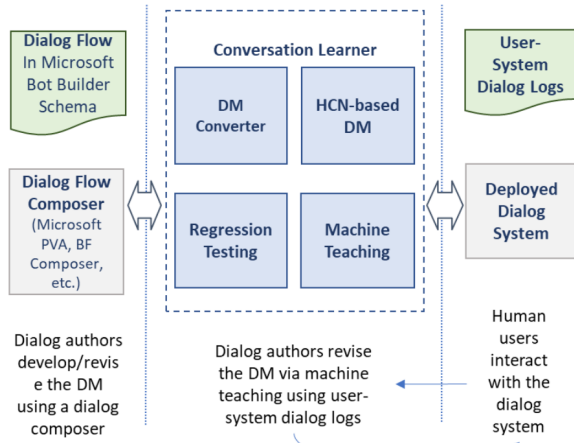[7] https://aws.amazon.com/lex/

Figure 2: The architecture of Conversation Learner (Top) and the development of DMs using Conversation Learner (Bottom).

that allows dialog authors to revise the HCN-based DM; and (4) an evaluation module that allows side-by-side comparison of the dialogs generated by different DMs. We describe each component in detail below.

2.1 HCN-based DM

The Conversation Learner HCN consists of a set of task-specific action templates, an entity module, a set of action masks, and a Recurrent Neural Network (RNN). Each action template can be a textual communicative action, rich card, or an API call. The entity module detects entity mentions in user utterances, grounds the entity mentions (e.g., by mapping an entity mention to a specific row in a dataset), and performs entity substitution in a selected action template to produce a fully-formed action (e.g., by mapping the template "the weather of [city]?" to "the weather of Seattle?"

Each action mask represents an "if-then" rule that determines the set of valid actions for some conditions (i.e., particular dialog states or user inputs).

The RNN maintains dialog states and selects system actions. For each turn in a training dialog, a combination of features, including the user utterance embedding, its bag of words vector, and the set of extracted entities are concatenated to form a feature vector that is passed to the RNN, specifically a Long Short Term Memory (LSTM) network. The RNN computes a hidden state vector, which is retained for the next timestep. Next, a softmax activation layer is used to calculate a probability distribution over the available system action tem-

plates. An action mask is then applied, and the result is normalized to select the highest-probability action as the best response for the current turn.

The HCN can be trained on a collection of user-system dialogs. For each system response in a dialog, the action template is labeled. The training of HCN takes two steps. First, all unique action templates are imported into the HCN. Then, the RNN, which maps states to action templates, is optimized for minimizing the categorical cross-entropy on training data. More specifically, each dialog forms one minibatch, and updates of the RNN are done via non-truncated back-propagation through time.

Readers are referred to Williams et al. (2017) for a detailed description of HCN. It should be noted that CL leverages the same network architecture as HCN with enhancements and modifications to generate context features from training samples.

2.2 DM converter

The DM converter converts a rule-based DM, developed using a dialog composer, to an HCN-based DM, which can then be improved via training dialogs and machine teaching.

Given a dialog flow, the DM converter automatically generates a set of training dialogs that represent the dialog flow. This process is done by performing an exhaustive set of walks over the dialog flow and generating training dialog instances for each walk. Rules that determine transitions in the dialog flow are represented as action masks in the HCN. The HCN is trained on the generated training dialogs as described in Section 2.1.

The DM converter can also convert a revised HCN-based DM back to a dialog flow by aggregating the individual training dialogs back into a graph for further editing and verification using a dialog composer.

2.3 Machine Teaching

The HCN-based DM can be improved via machine teaching (Simard et al., 2017). "Machine teaching" is an active learning paradigm that focuses on leveraging the knowledge and expertise of domain experts as "teachers". This paradigm puts a strong emphasis on tools and techniques that enable teachers - particularly non-data scientists and non-machine-learning experts - to visualize data, find potential problems, and provide corrections or additional training inputs in order to improve the system's performance.

For Conversation Learner, we developed a UI for visualizing and editing logged user-system dialogs that had failed to complete their tasks successfully. The teacher does not need to revise the DM directly (e.g., via writing code or by modifying dialog structure in a hierarchical composer tool). The teacher simply corrects cases where the dialog system responded poorly or incorrectly.

The teacher can make three types of corrections: (1) correct entity detection and grounding errors; (2) correct state-to-action mapping; or (3) create a new action template.

In cases where a large number of logged dialogs exists, we use active learning to provide a ranking of the candidate dialogs most likely to benefit from machine teaching intervention.

Our empirical tests show that machine teaching requires considerably fewer training samples than traditional machine learning approaches to improve system performance. We commonly observe significant improvements in DM performance by providing a dozen or fewer teaching examples.

There are three main reasons the HCN + machine teaching combination is so effective. First, dialog authors are generally subject matter experts who can make well-informed decisions about which actions the DM should perform in individual situations. If the DM has to automatically learn an action policy from logs, a large corpus of data is required. Second, the HCN allows dialog authors to explicitly encode domain-specific knowledge as action templates (bot activities or responses) and action masks (when a given response should be disallowed) without learning. Third, we can use intelligent filtering to select the most impactful failed dialogs for teachers to review.

2.4 Regression Testing

To effectively compare the performance of various dialog systems using different DMs, we developed a regression testing module. The module replays user utterances from transcripts of existing conversations against the DMs being tested; each DM then provides response action(s) for each turn. The module displays side-by-side comparisons of the resulting conversations from each DM, up to the point where the DM responses diverge. Human judges then rate the conversations as "left better","right better" or "same".

At the end of the rating session, a report is generated showing the performance of conversational flow amongst the DMs, as rated by the human judges.

3 System Demonstration

The system demonstration consists of the following steps:

1. The dialog author creates a rule-based DM using a dialog composer tool. We showcase the usage of Microsoft Power Virtual Agent system for this step.

2. The rule-based DM is exported to a common representation, e.g. Microsoft's '.dialog' schema[8].

3. The dialog flow is then imported into the Conversation Learner system. The import process involves traversing the dialog flow and creating dialog transcript instances for each traversal.

4. The generated dialog transcripts are run through both the rule-based system and Conversation Learner to demonstrate the improvement in dialog-handling flexibility provided by Conversation Learner's machine-learned DM.

5. For transcripts containing failures to complete the user's desired task, we demonstrate the machine teaching UI used to 'correct' errors and generate new training examples for revising the DM.

4 Case Study

To evaluate Conversation Learner's performance, we used dialog transcripts from Microsoft's text-based customer support system[9]. This customer support system uses a sophisticated rule-based DM to handle various customer support scenarios. It was created and is regularly updated by domain experts. We chose to use transcripts of actual user-system dialogs in order to measure the performance of the DMs in a realistic setting.

The customer support system consists of hundreds of "topics" out of which we selected three: "language settings", "installing or changing fonts",

---

[8]https://github.com/microsoft/botbuilder-dotnet/blob/master/schemas/sdk.schema

[9]These transcripts were scrubbed of any personally identifiable information.

Turn	Dialog
1	usr: How to reduce font size agt: How can I help you with fonts today?
2	usr: Adjust font size agt: Would you like to change the font size in an app or the size of text on your screen?
3	usr: Change size of text on screen agt: Change the size of text in Windows 10 using Display settings or use Magnifier. <Link to solution>
4	agt: Did that solve your problem? usr: Yes
5	agt: Great! Glad that helped.

Table 1: A sample human-agent dialog from the customer support logs

and "sign in problems". Table 1 shows an example of a user-system dialog for "change font" topic. Figure 3 shows the representation of a specific topic's dialog flow in the Microsoft Power Virtual Agents *dialog composer*. We exported the dialog flow graph of the support system from this system, then followed the process described in Section 2.2 to train our machine-learned HCN-based DM. Figure 4 shows an example of a generated train dialog.
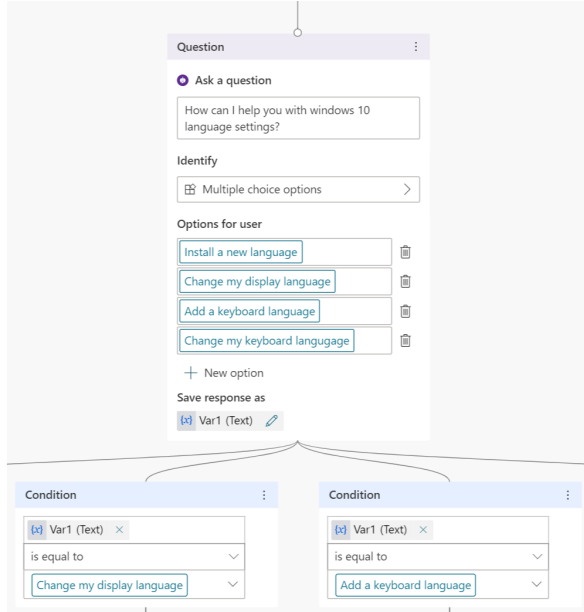
Figure 3: An example of a rule-based dialog defined in the Microsoft Power Virtual Agents system

After generating the HCN-based DM, we ran the set of dialog transcripts against both rule-based and HCN-based DMs. For the majority of conversations, users followed the expected flow, so the HCN-based DM produced the same results as the rule-based DM. For those that differed, we used human judges to do a blind qualitative evaluation of the conversations and choose the conversation

Figure 4: An example train dialog generated by traversing the different paths of a dialog tree (left), and DM actions generated from the tree (right)

that provided the best task-completion result.

User Rating	# of Convs.	%
CL is same	2749	91.63%
CL is better	136	4.53%
CL is worse	115	3.83%
Overall variation		0.7% (better)

Table 2: Initial results of human evaluation of 3000 dialogs against Conversation Learner (CL) and a rule-based dialog system.

As shown in Table 2, the HCN-based DM provided better results for many transcripts, but there was an almost equal number of dialogs where the rule-based DM was rated better. The rule-based DM may perform better in cases where specialized hard-coded logic was added to handle issues such as input normalization or rewriting.

An example dialog comparison is shown in Figures 5 and 6. As seen in Figure 6, the HCN-based DM handles cases where the user utterance does not match a known phrase, better than the rules-based system. The HCN is also able to handle unexpected transitions between dialog nodes.

In a rule-based system, updating the DM to handle unexpected user responses and off-track conversations is much harder due to the rigid structure of the dialog flow graph, the long-tail nature of user-system dialog transcripts (a large number of sparse examples), and complexity in transitioning between unrelated parts of the dialog flow.

Next, we demonstrate that the performance of the HCN DM can be substantially improved by adding just 3 to 5 teaching examples via the machine teaching UI presented in Section 2.3. For this experiment, we chose dialog transcripts that contained common patterns of conversational problems, like users switching context, repeating themselves or asking follow-up questions, and corrected
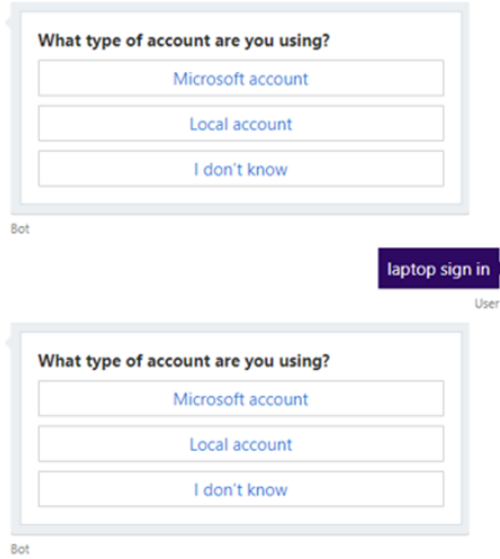
Figure 5: A sample dialog from a rule-based system. Notice that the system just repeats its previous question (as a rule) since it did not understand the user reply.
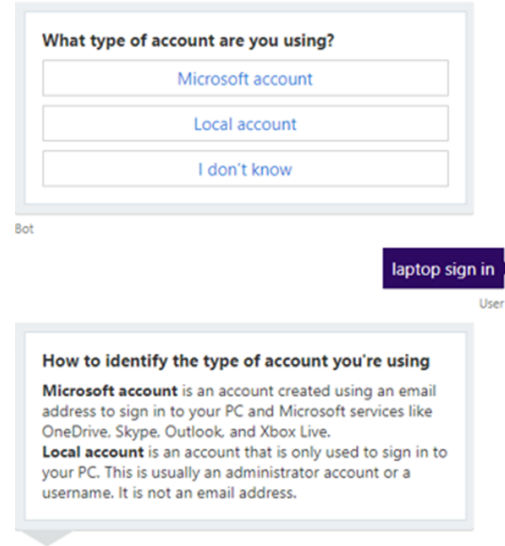

Figure 6: Same dialog as Figure 5 in Conversation Learner. Notice that the user response is accurately generalized to one of the available options when possible.
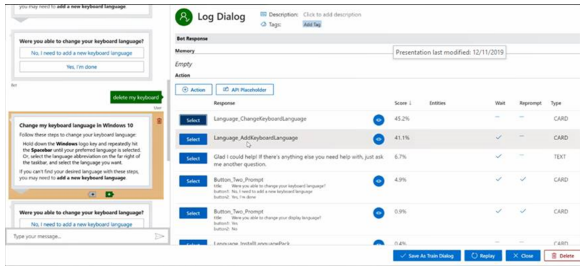

Figure 7: Machine Teaching UI for *correcting* dialogs to revise DM.

the dialog policy by creating or selecting the appropriate system action to resolve the problem. Once these additional examples are added, as illustrated in Figure 7 the HCN-based DM's performance improvement over the rule-based DM nearly tripled, from 4.53% to 13.8%.

User Rating	# of Convs.	%
CL is same	2562	85.4%
CL is better	414	13.8%
CL is worse	24	0.81%
Overall variation		12.99% (better)

Table 3: Results of human evaluation of 3000 dialogs after improving Conversation Learner (CL) model with machine teaching.

As shown in Table 3, minimal intervention from a dialog author by providing a small number of corrections to problematic user-system dialog logs

can have a significant impact on the performance of the DM. As new users interact with the system and new transcripts are generated, the dialog author can continuously improve the HCN DM's performance by making corrections and adding new training data.

5 Conclusion

In this paper, we presented *Conversation Learner*, a machine teaching tool for building dialog policy managers. We have shown that the CL HCN-based DM can be bootstrapped from a rule-based DM preserving the same behavior expected from the rule-based system. Using the CL machine teaching UI, the dialog author can provide corrections to the logged user-system dialogs and further improve the CL's DM performance. We demonstrated this through a case study based on dialog transcripts from Microsoft's text-based customer support system where the gains were approximately 13%.

We are planning to extend this work by looking into following problems: 1) Investigating effectiveness of different ranking algorithms for log correction recommendation, 2) Optimizing number of training samples and action masks generated from the rule-based DM, and 3) Improving predictions of HCN-based DM by looking into alternative network architectures.

References

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *CoRR*, abs/1712.05181.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia. Association for Computational Linguistics.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Xiang Li, Yaoqin Zhang, Zheng Zhang, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, and Jianfeng Gao. 2019. Convlab: Multi-domain end-to-end dialog system platform. *CoRR*, abs/1904.08637.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Melbourne, Australia. Association for Computational Linguistics.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2Seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1468–1478, Melbourne, Australia. Association for Computational Linguistics.

Alexander H. Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. *CoRR*, abs/1705.06476.

Alexandros Papangelis, Mahdi Namazifar, Chandra Khatri, Yi-Chia Wang, Piero Molino, and Gokhan Tur. 2020. Plato dialogue system: A flexible conversational ai research platform.

Patrice Y. Simard, Saleema Amershi, David Maxwell Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and John Wernsing. 2017. Machine teaching: A new paradigm for building machine learning systems. *CoRR*, abs/1707.06742.

Stefan Ultes, Lina M. Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, and Steve Young. 2017. PyDial: A Multi-domain Statistical Dialogue System Toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78, Vancouver, Canada. Association for Computational Linguistics.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *CoRR*, abs/1702.03274.