SkillBot: Towards Automatic Skill Development via User Demonstration

Yilin Shen, Avik Ray, Hongxia Jin, Sandeep Nama
Samsung Research America, Mountain View, CA, USA
{yilin.shen,avik.r,hongxia.jin,s.nama}@samsung.com

Abstract

We present SkillBot that takes the first step to enable end users to teach new skills in personal assistants (PA). Unlike existing PA products that need software developers to build new skills via IDE tools, an end user can use SkillBot to build new skills just by naturally demonstrating the task on device screen. SkillBot automatically develops a natural language understanding (NLU) engine and implements the action without the need of coding. On both benchmark and in-house datasets, we validate the competitive performance of SkillBot automatically built NLU. We also observe that it only takes a few minutes for an end user to build a new skill using SkillBot.

1 Introduction

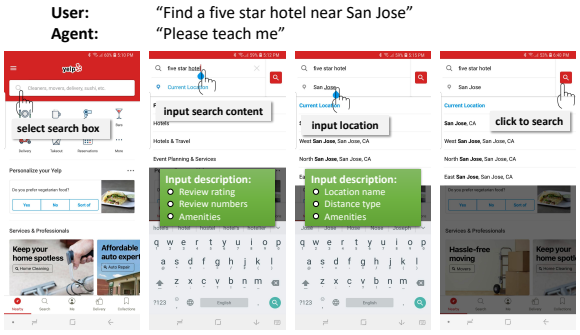
Artificially intelligent voice-enabled personal assistants (PA) have been emerging in our daily life, such as Alexa, Google Assistant, Siri, Bixby, etc. Existing off-the-shelf personal assistants serve different domains, in which each domain has a large number of capabilities, called *skills*. A skill refers to the understanding of various utterances about one intent/task and the execution of this intent/task.

Existing industrial PA products completely rely on software developers to build new skills by manually developing NLU engine and implementing action fulfillment. On one hand, while recent work CRUISE (Shen et al., 2018a) and SliQA-I (Shen et al., 2019) have introduced automatic training utterance and question generation approaches with lightweight human workload, they still require the involvement of software developers for NLU development through IDE tools. Another line of research is to personalize NLU engine in existing skills (Azaria et al., 2016; Ray et al., 2018; Shen et al., 2018b; Wang et al., 2018a), yet they cannot support building new skills. On the other hand, developers need to write a significant amount of code in order to fulfill a sequence of operations to carry out the task (DialogFlow, 2018; Alexa, 2018).

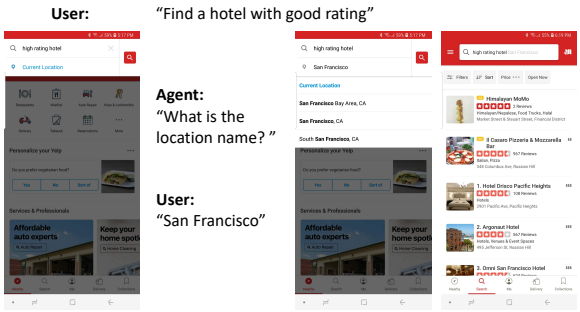
However, in practice, it is infeasible for software developers to pre-build all possible skills that satisfy all users' needs. First, the pre-collected training corpus usually cannot exhaustively cover all possible varieties of utterances. Second, due to the heavy workload on fulfillment implementation, many actions are not implemented to be supported in PA. Thus, it is critically desirable to design an easy to use system that can facilitate *end users* to quickly build high quality new skills. Unlike existing IDE for skill development, such system needs to be more friendly and natural to ordinary end users without any complex interfaces.

In this paper, we take the first step to present SkillBot that enables end users to initialize building PA skills. An ordinary user, without either natural language expertise or software development background, only needs to demonstrate the task on his daily device screen. SkillBot, without a complex IDE interface, automatically learns the action by tracking user operations and develops the NLU engine by automatically generating training utterances based on pre-built skills. Since users intend to use spoken language to interact with PA, we follow most industrial products (DialogFlow, 2018; Alexa, 2018) to use spoken language understanding (SLU) as our NLU engine, which understands user query by detecting its intent (skill) and extracting semantic slots (slot filling) (Liu and Lane, 2016; Kim et al., 2017; Wang et al., 2018b). Even though our fully automated SkillBot only aims to satisfy each user's personal needs rather than understand all various expressions

Proceedings of NAACL-HLT 2019: Demonstrations, pages 105–109
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) Learning via Demonstration: When PA asks for teaching, the user operates step by step accordingly in Yelp app. For each user input, SkillBot prompts to ask user to select the most relevant description of this input.



(b) Automatic Execution: After parsing the utterance using NLU, PA executes the learned action step by step. When finding a missing input, PA prompts for user input.

Figure 1: SkillBot Running Example: Learning and executing new skill "find hotel" in Yelp

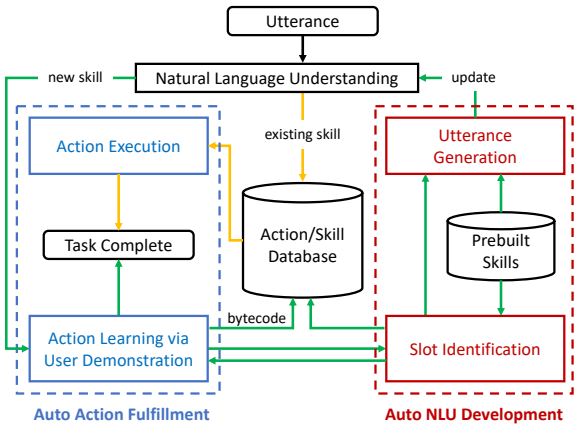


Figure 2: SkillBot Architecture & Workflow: When learning via demonstration (green arrows), automatic action fulfillment (runs as a system service on each user's device) first tracks and captures system level event sequence based on user operations. It outputs a bytecode file with learned event sequence for this skill and saves in database for future execution. After each user input operation, automatic NLU development identifies the list of possible slot descriptions for user to select and then generates more training utterances by leveraging existing training utterances in pre-built skills. At last, the NLU engine is updated using all generated utterances. When executing an existing skill (orange arrows), based on the parsed skill from NLU, the corresponding bytecode is retrieved to automatically execute all saved steps.

from any user, we show the competitive NLU performance of SkillBot built skills in later experiments. More importantly, in most cases it only takes a user several minutes to build a new skill using SkillBot. As such, SkillBot leverages the existing skills to help end users automatically and quickly build high quality skills to meet their personal needs.

2 SkillBot System Overview

2.1 Our Settings

To satisfy user personalized needs in PA, SkillBot aims to enable end users to build their own (long-tail) skills *only by demonstration on screen*. That said, an end user only *naturally* uses their device as usual to build a new skill. In this first work, we assume that there exist a set of pre-built popular (head) skills in the ecosystem and these skills contain both annotated training utterances and a text description for each slot. In addition, we target on teaching and executing actions on the same mobile apps.

Figure 1 shows a running example in which SkillBot helps an end user to build the new "find hotel" skill in Yelp. For an end user, he uses PA as usual via voice utterances. SkillBot prompts the user to teach when PA cannot understand and execute the user utterance correctly. As in Figure 1a, all the end user needs to do is to demonstrate on screen step by step how he wants PA to execute. After each user input, SkillBot identifies possible slot descriptions and asks user to select the most relevant one. SkillBot then automatically builds the new skill and outputs a well-trained NLU engine and an action executable file. A user could teach a new skill multiple times where each skill is considered to have the same on-screen operation sequence. Next time, in Figure 1b, PA can correctly understand this user's different expressions of this intent and execute the right action.

2.2 System Design

Recall that our target is to facilitate the end users who have neither natural language expertise nor software development knowledge. Thus, SkillBot is designed to support these two automation respectively. Specifically, SkillBot consists of two main components, *automatic action fulfillment*

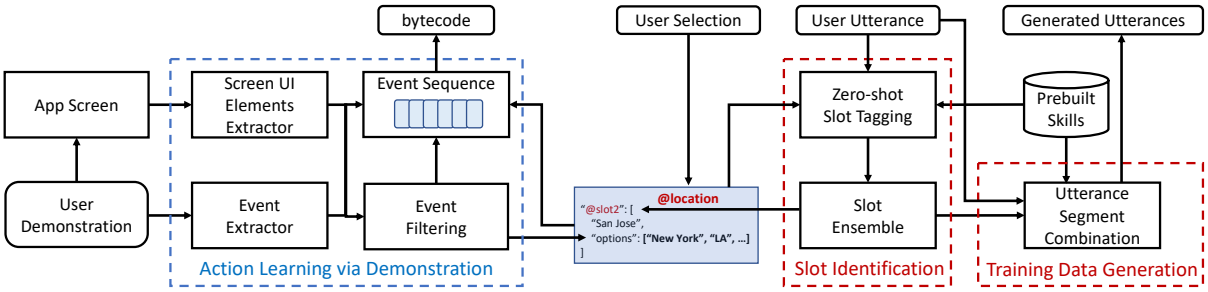


Figure 3: New Skill Learning in SkillBot

and *automatic NLU development*. In addition, SkillBot also has an action/skill database to save the mapping between learned actions and corresponding skills. Figure 2 shows SkillBot architecture and work flow. Since we use the off-the-shelf NLU models in this paper, auto NLU development focuses on generating annotated training utterances.

3 Learning via Demonstration

3.1 Action Learning

Action learning module has two main threads, as shown in the left part of Figure 3. Following user demonstration, at each screen, screen UI element extractor collects all UI elements in the format of a DOM tree on the current screen that the user is operating. In the meanwhile, event extractor collects all events from user operates in this screen.

Since there are typically a lot of system services running on device, the extracted events usually include many irrelevant ones that is not from user demonstrated actions, such as 'windows change', 'window state change', 'system or other app notifications', etc. In order to filter the irrelevant events, we first prune all events out of the current front end app based on their event package name. To further ensure some unexpected events within app (e.g., location permission request in Yelp app), we allow user to teach again at any point when user sees any unusual popups or notifications. At last, a bytecode is outputted including the event sequence in which each event contains its UI element information and the required slot value input based on the identified slot in Section 3.2.1.

3.2 Automatic NLU Development

The key idea of training utterance generation is to identify the slots in the user utterance and then use them to generate more training utterances based on the training utterances in existing pre-built skills.

3.2.1 Slot Identification

This module is invoked after each user input during demonstration. As shown in the middle part (blue) of Figure 3, after user inputs "San Jose" in the example of Figure 1a, it receives the user utterance and the optional values (e.g., "New York", "LA", etc.) extracted from dropdown list of the user input box (Yelp location textbox) during action learning.

Taking the above input, we first construct the set of natural language utterances by replacing the input values with other optional values. For each utterance, zero-shot slot tagging module extracts its semantic slots based on each slot description using the zero-shot model in (Bapna et al., 2017) trained on pre-built skills. Slot ensemble module performs a joint slot detection across all constructed utterances by combining the likelihood scores of each slot. The descriptions of identified top ranked slots are sent to the user to select the most relevant one.

3.2.2 Utterance Generation

In this first work, we assume that each training utterance in pre-built skills has been decomposed into segments by human expert or our proposed CRUISE approach (Shen et al., 2018a). Each segment contains a slot tag (two examples are shown in the right side of Figure 4). We only use the subset of segments associated with the aforementioned identified slots.

We generate the utterance by combining identified segments into long utterances by concatenating them together (middle part in Figure 4). To do so, we first use the off-the-shelf Stanford parser to identify the verb and main object in user utterance. In the sample utterance of Figure 1, "find" and "hotel" are marked as verb and object based on the parser tree. As

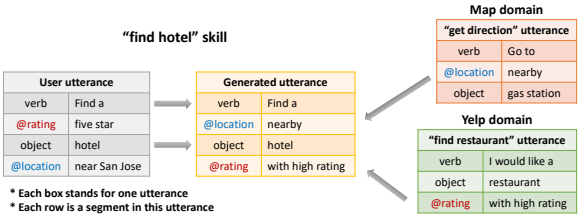


Figure 4: Utterance Generation Example

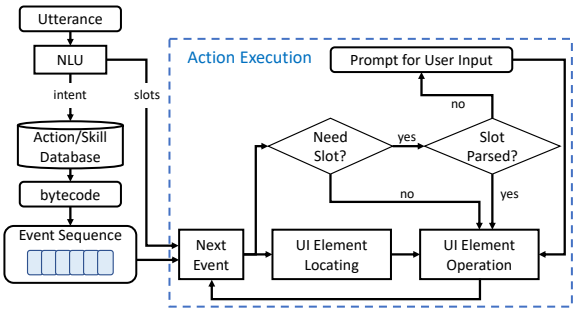


Figure 5: Automatic Skill Execution in SkillBot

the arrows show in Figure 4, we next generate the permutations of segments before and after the object based on their places in original utterances with non-overlapping slots.

4 Automatic Execution

Figure 5 shows the flow of automatic execution of a skill. NLU first parses the utterance and outputs its intent and slots. The intent class is used to query the action/skill database to retrieve the bytecode of the corresponding action. SkillBot executes the events one by one following the sequence. For each event, the execution consists of the following two threads: One thread determines if this event requires a slot input based on the saved meta data during learning. If so, we extract on the slot results parsed by NLU. If this information is missing in utterance or NLU fails to parse, we prompt the follow-up question to ask user. The other thread first locates the UI element based on its saved coordinates. It then inputs slot values and simulates the user operation by using gesture control based on the element coordinates (e.g., MotionEvent in Android devices).

5 Experiments

In this section, we focus on evaluating SkillBot built NLU engine given that auto action learning and execution are restricted to the same app.

We test on both benchmark and in-house datasets/domains: (1) *ATIS* (4978 training, 893 test) with 17 intent labels and 79 slot labels (Hemphill et al., 1990); (2) *In-house Yelp* (1968 training, 911 test) with 5 intents and 10 slots. To evaluate both datasets which are not generated by CRUISE, we segment each utterance using dependency parser and slot annotations (each segment ends with a slot). The noisy segments are further removed by human experts. In the experiment, we assume that user always select the correct slot description after each input to ensure the correct slot identification. We use the benchmark BiRNN based joint NLU model (Liu and Lane, 2016).

In each dataset (with k intents), for each intent I, we assume the user teaches this intent I given the remaining $k-1$ intents as pre-built skills. Let $\mathcal{T}_I$ be the subset of training data w.r.t. intent I, and $\mathcal{T}_I^c$ be utterances w.r.t. the remaining $k-1$ intents. Our baseline NLU engine is trained on this training set $\mathcal{T}_I^c$ of remaining $k-1$ intents. Next, we randomly sample 5 user utterances (assuming a user teaches the new intent 5 times) from set $\mathcal{T}_I$, and use our utterance generation algorithm to auto generate dataset A_I for intent I. Our SkillBot built NLU (called *SkillBot NLU*) is trained on the combined training data $\mathcal{T}_I^c \cup A_I$. Both the NLU engines are tested on the original test dataset (having all k intents).

Table 1 presents the results. For ATIS, we show the results for top 8 intents, which contain at least 0.9% of the overall training utterances (the third column displays this fraction). SkillBot NLU achieves a large gain in intent accuracy over the baseline in both datasets. Moreover, we observe that the accuracy gain is also roughly correlated to the fraction of all test utterances in each intent. This indicates that the SkillBot NLU can correctly learn most of the test utterances in the newly added intent. SkillBot NLU also improves the slot filling F1 score in most intents. In cases when SkillBot NLU completely misses a slot, it can directly ask a follow-up question to allow user provide the slot value before action fulfillment. Therefore, SkillBot with follow-up user clarification further improves the F1 score and obtains performance gain in last column.

6 Discussion & Future Work

We present the first SkillBot that enables end users to build skills in PA. SkillBot automates both action fulfillment and NLU development. In

Table 1: SkillBot NLU Experimental Results

Domain	Intent/Skill	Train	Intent (Accuracy)			Slot Filling (F1 Score)			
		Size(%)	Baseline	SkillBot	Gain (%)	Baseline	SkillBot	SkillBot+Follow-up	Gain (%)
ATIS	atis_flight	74.34	27.00	**60.40**	33.40	85.13	**87.45**	**89.37**	4.24
	atis_airfare	8.59	90.40	**92.40**	2.00	97.43	96.96	97.39	-0.04
	atis_ground_service	5.13	92.40	**97.20**	4.80	96.69	**96.99**	**97.36**	0.67
	atis_airline	3.14	94.00	**96.40**	2.40	97.43	**97.54**	**97.98**	0.55
	atis_abbreviation	2.90	94.20	**95.60**	1.40	97.04	**97.45**	**97.88**	0.84
	atis_aircraft	1.56	94.80	**96.40**	1.60	97.54	97.46	**97.92**	0.38
	atis_flight_time	1.00	95.40	**97.60**	2.20	97.07	**97.37**	**98.06**	0.99
	atis_quantity	0.91	95.40	**97.20**	1.80	97.37	97.05	**97.56**	0.19
Yelp	search_restaurant	23.23	78.16	**88.04**	9.88	92.50	**92.85**	**94.71**	2.21
	get_directions	24.20	72.78	**85.40**	12.62	93.29	92.10	**94.36**	1.07
	bookmark	22.75	76.73	**84.96**	8.23	94.13	**94.90**	**96.90**	2.77
	reservation	12.53	89.13	**95.06**	5.93	93.53	**94.06**	**96.15**	2.62
	call_restaurant	16.60	82.66	**92.54**	9.88	93.31	92.12	**94.46**	1.15

future work, we will evaluate SkillBot in more real and larger-scale scenarios: allow end users to teach more naturally with less clarification by improving the accuracy of slot identification; support building a brand new domain by enabling slot identification to map more varieties of slots in other domains; incorporate with other non-CRUISE pre-built skills without pre-segmented utterances; enhance the robustness of action learning to tackle the dynamics in mobile device system; expand cross app action teaching in which we will design machine learning algorithms for event filtering and UI semantic mapping.

Acknowledgments

The authors would like to thank Abhishek Patel and Xiangyu Zeng for useful discussion and their help of demo implementation and video shooting.

References

Amazon Alexa. 2018. https://developer.amazon.com/docs/custom-skills/handle-requests-sent-by-alexa.html.

Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. 2016. Instructable intelligent personal agent. In *AAAI*, pages 2681–2689.

Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. In *Interspeech*, pages 2476–2480.

DialogFlow. 2018. https://dialogflow.com/.

Charles T Hemphill, John J Godfrey, George R Doddington, et al. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101.

Young-Bum Kim, Sungjin Lee, and Karl Stratos. 2017. ONENET: joint domain, intent, slot prediction for spoken language understanding. In *ASRU*, pages 547–553.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Avik Ray, Yilin Shen, and Hongxia Jin. 2018. Learning out-of-vocabulary words in intelligent personal agents. In *IJCAI*, pages 4309–4315.

Yilin Shen, Avik Ray, Abhishek Patel, and Hongxia Jin. 2018a. CRUISE: cold-start new skill development via iterative utterance generation. In *ACL, System Demonstrations*, pages 105–110.

Yilin Shen, Yu Wang, Abhishek Patel, and Hongxia Jin. 2019. Sliqa-i: Towards cold-start development of end-to-end spoken language interface for question answering. In *ICASSP*.

Yilin Shen, Xiangyu Zeng, Yu Wang, and Hongxia Jin. 2018b. User information augmented semantic frame parsing using progressive neural networks. In *Interspeech*, pages 3464–3468.

Yu Wang, Abhishek Patel, Yilin Shen, and Hongxia Jin. 2018a. A deep reinforcement learning based multimodal coaching model (dcm) for slot filling in spoken language understanding(slu). In *Interspeech*, pages 3444–3448.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018b. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *NAACL-HLT*, pages 309–314.