Dialog for Language to Code

Shobhit Chaurasia and **Raymond J. Mooney**
Department of Computer Science
The University of Texas at Austin
shobhit@utexas.edu, mooney@cs.utexas.edu

Abstract

Generating computer code from natural language descriptions has been a long-standing problem. Prior work in this domain has restricted itself to generating code in one shot from a single description. To overcome this limitation, we propose a system that can engage users in a dialog to clarify their intent until it has all the information to produce correct code. To evaluate the efficacy of dialog in code generation, we focus on synthesizing conditional statements in the form of IFTTT recipes.

1 Introduction

Building a natural language interface for programmatic tasks has long been a goal of computational linguistics. This has been explored in a plethora of domains such as generating database queries (Zelle and Mooney, 1996; Berant et al., 2013; Yin et al., 2016), building regular expressions (Manshadi et al., 2013), commanding a robot (She et al., 2014), programming on spreadsheets (Gulwani and Marron, 2014), and event-driven automation (Quirk et al., 2015), each with its own domain-specific target language. Synthesis of computer programs in general-purpose programming languages has also been explored (Ling et al., 2016; Yin and Neubig, 2017). The existing work assumes that a working program can be generated in one shot from a single natural language description. However, in many cases, users omit important details that prevents the generation of fully executable code from their initial description.

Another line of research that has recently garnered increasing attention is that of dialog systems (Singh et al., 2002; Young et al., 2013). Dialog systems have been employed for goal-directed tasks such as providing technical support (Lowe et al., 2015) and travel information and booking (Williams et al., 2013), as well as in non-goal oriented domains such as social-media chat-bots (Ritter et al., 2011; Shang et al., 2015).

In this paper, we combine these two lines of research and propose a system that engages the user in a dialog, asking questions to elicit additional information until the system is confident that it fully understands the user's intent and has all of the details to produce correct, complete code. An added advantage of the dialog setting is the possibility of continuous improvement of the underlying semantic parser through conversations (Artzi and Zettlemoyer, 2013; Thomason et al., 2015; Weston, 2016), which could further increase success rates for code generation and result in shorter dialogs. We focus on a restrictive, yet important class of programs that deal with conditions, i.e., `if-then` statements. To this end, we use the IFTTT dataset released by Quirk et al. (2015). To the best of our knowledge, this is the first attempt to use dialog for code generation from language.

2 Task Overview

2.1 IFTTT Domain

IFTTT (if-this-then-that) is a web-service that allows users to automate simple tasks by creating short scripts, called *recipes*, through a GUI that enables them to connect web-services and smart devices. A recipe consists of a *trigger* — an event which fires the recipe — and an *action* — the task to be performed when the recipe is fired. A trigger is characterized by a *trigger channel* (the source of the event) and a *trigger function* (the nature of the event); an action is characterized by an *action channel* (the destination of the task to be performed) and an *action function* (the nature of that task). Users can share their recipes publicly with short descriptions of their functionalities.

For example, a recipe with description "Text me when I am tagged in a picture on Facebook" might have trigger channel `facebook`, trigger function `you_are_tagged_in_a_photo`, action channel `sms`, and action function `send_me_an_sms`.

2.2 Problem Statement

Our goal is to synthesize IFTTT recipes from their natural language descriptions. Unlike prior work in this domain (Quirk et al., 2015; Dong and Lapata, 2016; Beltagy and Quirk, 2016; Liu et al., 2016), which restrict the system to synthesizing a recipe from a single description, we seek to enable the system to interact with users by engaging them in a dialog to clarify their intent when the system's confidence in its inference is low. This is particularly crucial when there are multiple channels or functions achieving similar goals, or when the initial recipe descriptions are vague.

3 Approach

We propose a dialog system with which users can converse using natural language to create recipes. It consists of three components: Dialog Manager, Natural Language Understanding (NLU), and Natural Language Generation (Jurafsky, 2000).

3.1 Dialog Manager

The aim of the dialog system is to determine values of channels and functions for the recipe that the user wants to create. We cast this problem as a slot-filling task, in which the system maintains a belief state — its current estimates for the slots — and follows a hand-coded policy to update its belief state until it is confident that the belief state is same as the user goal. The strategy is similar to the one used by Thomason et al. (2015).

3.1.1 Belief State

The belief state consists of four slots: `trigger_channel`, `trigger_function`, `action_channel`, and `action_function`. The slots naturally form a hierarchy: channels are above functions. Although triggers and actions are, in a loose sense, at the same level in the hierarchy[1], it is more natural to specify triggers before actions, thereby inducing a complete hierarchy over slots. This hierarchy is exploited in specifying a policy for the dialog system.

[1]Technically, the presence of *ingredients* — properties associated with trigger functions that can be utilized by action functions — puts triggers above actions in the hierarchy.

The system maintains a probability distribution over all possible values for each slot. After each user utterance, the probability distribution for one or more slots is updated based on the parse returned by the utterance parser (see Section 3.2). The system follows a hand-coded policy over the discrete state-space obtained from the belief state by assigning the values with highest probability (candidates with highest confidence) to each slot.

3.1.2 Static Dialog Policy

The dialog opens with an open-ended user utterance (a user-initiative) in which the user is expected to describe the recipe. Its parse is used to update all the slots in the belief state. The system moves down the slot-hierarchy, one slot at a time, and picks the next action based on the confidence of the top candidate for each slot. If the confidence is above α, the parse is accepted, and the candidate is assigned to that slot. If the confidence is below β, the parse is rejected, and the system requests information for that slot (a system-initiative). If the confidence is between α and β, the system seeks a confirmation of the candidate value for that slot; if the user affirms, the candidate is assigned to the slot, otherwise the system requests information for that slot. Value of α and β present a trade-off between dialog success and dialog length. $\alpha = 0.85$ and $\beta = 0.25$ were used in all the experiments, chosen by analyzing the performance of the dialog system on the IFTTT validation set.

3.2 Natural Language Understanding

This component is responsible for parsing user utterances. We use the model proposed by Liu et al. (2016): an LSTM-based classifier enhanced with a hierarchical, two-level attention mechanism (Bahdanau et al., 2014). In our system, the semantic parser is composed of a set of four such models, one for each slot. User-initiatives are parsed by all four models, while user responses to system-initiatives are parsed by the model corresponding to the slot under consideration.

3.3 Natural Language Generation

The dialog system uses templates and IFTTT API documentation to translate its belief state into a comprehensible utterance. For example, the confirmation request for the `blink_lights` action function of the `hue` action channel is: "Do you want to *briefly turn your hue lights off then back on* every time the applet is triggered?"

3.4 Retraining NLU using Dialog

Another advantage of using a dialog approach to recipe synthesis is that it unlocks the possibility of continuous parser improvement through conversations (Artzi and Zettlemoyer, 2013; Thomason et al., 2015; Weston, 2016). To this end, we extract training data from the dialogs. Opening user utterances and user utterances for each slot after a system-initiative in successful dialogs are paired with inferred slot values to retrain the models. Analysis of models' predictions on the validation set revealed that the attention mechanism was rather good at attending to relevant parts of an utterance; the models failed because they often couldn't pick the correct channel or function among the similar ones. Therefore, we tuned only the non-attention parameters during retraining.

4 Experiments

We trained our parser on the training set of the IFTTT corpus. Since the corpus was released, many recipes have been taken down; we could only obtain $66,588$ out of $77,495$ training recipes. We evaluated our system on the "gold" subset of the IFTTT test set created by Quirk et al. (2015) which consists of 550 recipes on which at least three humans presented with the recipe descriptions agreed with the true labels. We restricted ourselves to this subset because our experiments involved humans interacting with the dialog system to describe the recipe and answer its questions, and it was crucial that they themselves have a clear understanding of the recipe.

4.1 Experimental Setup

We used Amazon Mechanical Turk to conduct our experiments. Users were provided with recipe descriptions from the IFTTT corpus. Since descriptions could often be vague, we also explicitly provided them the details of channels and functions associated with the recipes. As noted in (Thomason et al., 2015), in which a similar interface was used to let users chat with a robot for performing navigation- and delivery-related tasks, information presented using words could linguistically prime the users. They avoided priming completely by presenting the information pictorially. Since it is unclear how to succinctly describe a recipe without using words, we, instead, used keywords for channels and functions (such as `event_starts` and `blink_lights`), which usually contain only content words necessary to give an indication of their functionality, but are somewhat distant from natural language. Additionally, we encouraged users to use their own words based on their understanding of the recipe and restricted direct usage of these keywords. Fig. 1 shows a sample dialog.

ROBOT: Hi! Please describe the applet you want to create for automating the task you have on your mind.
YOU: Notify Glass to take and share pictures for Events.
ROBOT: Do you want an event on the Google Calendar service to trigger the applet?
YOU: yes
ROBOT: Which event on the Google Calendar service should cause the applet to run?
YOU: any event that will start
ROBOT: The applet will trigger within 15 minutes of the starting time of any event on your google calendar. It will use the Google Calendar service to look for this event. The action taken will be to send a notification to your google glass timeline. This action will be performed using the Google Glass service. Is this what you wanted?
YOU: yes
ROBOT: Ok, bye!

Figure 1: Dialog for a recipe with trigger `event_starts` on `google_calendar`, action `send_notification` on `google_glass`.

4.2 Dialog Experiments

We conducted two experiments to evaluate the efficacy of dialog in synthesizing recipes. In both the experiments, two baselines are used. First is the the best-performing model from Liu et al. (2016), currently the state-of-the-art on this task, provided only with initial recipe descriptions, as should be the case for a single-shot model. The second baseline, called "Concat," uses the same model as above, but is provided with all the user utterances from the conversation concatenated. By ensuring that both the single-shot and the dialog approach get same information, the Concat baseline provides a middle-ground between the two approaches, and is more fair to the single-shot system, but disguises its obvious deficiency: the lack of ability to ask for clarification.

4.2.1 Constrained User-Initiative

To evaluate our system directly on the test set, we constrained the users to use the original recipe descriptions as their first utterance (i.e. the user-initiative) when they were asked by the system to describe the recipe. This way, we can directly compare our results with prior work which uses this set for evaluation.

4.2.2 Free User-Initiative

To emulate a more realistic setting in which users drive the entire conversation, including the user-initiative, we allowed the users to provide the ini-

Experiment	Liu et al. (2016)	Concat Baseline	Ours	
	Accuracy	Accuracy	Accuracy	Avg. dialog length
Constrained UI	85.28[2]	91.27	95.28	2.55
Free UI	66.0	77.82	81.45	4.04
After retraining	66.0	77.48	82.55	4.08

Table 1: Accuracy of recipe synthesis. Average dialog length is measured in terms of number of user utterances.

tial recipe descriptions themselves. For a fair comparison, we evaluated the Liu et al. (2016)'s baseline model on the initial descriptions provided in the conversations.

4.3 Results

The results are summarized in Table 1. The dialog approach boosts the accuracy of recipe synthesis considerably over the single-shot approach in both the experiments: 10 point increase with constrained user-initiative and over 15 point increase with free user-initiative. Even when the two approaches receive the same information (i.e., when the dialog approach is compared with the Concat baseline), the dialog approach boosts the accuracy by approximately 4 points.

Surprisingly, the accuracy of both the single-shot approach and the dialog approach fell dramatically in the experiment with free user-initiative. We contend that the reason behind this reduction is the difference between the two settings in which the recipe descriptions were created: Constrained UI experiment uses original descriptions written by the authors of the recipes with the aim of summarizing their recipes so that their functionality can be easily understood by others *without their assistance*. The descriptions used in Free UI experiment were provided by humans with the aim of describing the recipe to a system with the knowledge that the system can ask clarification questions. The former are expected to be more descriptive and self-contained than the latter. The larger average dialog length in the Free UI experiment further corroborates this point.

4.4 Parser Retraining

Parser retraining would be most helpful when the data is extracted from conversations that involve channels and functions for which the existing parser's confidences are low. Therefore, we randomly sampled 100 recipes from an unused portion of the test set on which the confidence of existing parser is below β for at least two slots. About 130 data-points were extracted from conversations with humans over these recipes, and the four models were retrained.

4.4.1 Results

The accuracy of systems using retrained models is summarized in Table 1. For direct comparison, the dialog system with retrained models was evaluated using the user utterances from conversations in the Free UI experiment, except when its actions deviated from the original ones — due to an improved NLU component — in which case new user utterances were obtained. While retraining didn't improve the single-shot accuracy, there was a marginal improvement of 1.1 points in the dialog setting. Analysis of the conversations revealed that this was because the retrained models had lower confidence for some channels and functions for which it initially had high priors. On one hand, this helped the dialog system avoid getting stuck in an impasse when it assigns an incorrect value to a slot with high confidence without confirmation. On the other hand, this pessimism led to a slight increase in average dialog length.

5 Future Work

In this work, we focus only on conditionals. A natural extension would be to consider other programming constructs such as loops, procedure invocations, and sequence of execution. Dialog policy learning can be added to account for non-stationarity in the dialog environment due to parser learning (Padmakumar et al., 2017).

6 Conclusion

In this work, we demonstrated the efficacy of using dialog for mapping natural language to short, exe-

[2]The accuracy reported by Liu et al. (2016) is 87.5%. Our implementation of their system was able to achieve only 85.28% accuracy. The discrepancy could be because of a smaller training set (they had 68k recipes), a smaller gold test set (they had 584 recipes), or variance while training.

cutable computer code. We evaluated this idea in the domain of IFTTT recipes. The proposed system engaged the user in a dialog, asking questions to elicit additional information until it was confident in its inference, thereby increasing the accuracy on this task over the state-of-the-art models that are restricted to synthesizing recipes in one shot by $10 - 15$ points. Additionally, we demonstrated how data extracted from the conversations can be used for continuous parser learning.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback, and Aishwarya Padmakumar for her ideas and insights. This work was supported by a grant from Microsoft Research.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1(1):49–62. https://homes.cs.washington.edu/ lsz/papers/aztacl13.pdf.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

I. Beltagy and Chris Quirk. 2016. Improved semantic parsers for if-then statements. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 726–736. http://www.aclweb.org/anthology/P16-1069.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*. http://www.aclweb.org/anthology/D13-1160.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 33–43. http://www.aclweb.org/anthology/P16-1004.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. ACM, SIGMOD '14, pages 803–814. http://doi.acm.org/10.1145/2588555.2612177.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 599–609. http://www.aclweb.org/anthology/P16-1057.

Chang Liu, Xinyun Chen, Eui Chul Shin, Mingcheng Chen, and Dawn Song. 2016. Latent attention for if-then program synthesis. In *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 4574–4582. http://papers.nips.cc/paper/6284-latent-attention-for-if-then-program-synthesis.pdf.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 285–294. http://aclweb.org/anthology/W15-4640.

Mehdi Manshadi, Daniel Gildea, and James Allen. 2013. Integrating programming by example and natural language programming. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'13, pages 661–667. http://dl.acm.org/citation.cfm?id=2891460.2891552.

Aishwarya Padmakumar, Jesse Thomason, and Raymond J. Mooney. 2017. Integrated learning of dialog strategies and semantic parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*. http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=127615.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (July):878–888. http://www.aclweb.org/anthology/P15-1085.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, EMNLP '11, pages 583–593. http://dl.acm.org/citation.cfm?id=2145432.2145500.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages

1577–1586. http://www.aclweb.org/anthology/P15-1152.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Association for Computational Linguistics, pages 89–97. http://www.aclweb.org/anthology/W14-4313.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research* 16:105–133. https://www.jair.org/media/859/live-859-1983-jair.pdf.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, IJCAI'15, pages 1923–1929. http://dl.acm.org/citation.cfm?id=2832415.2832516.

Jason E Weston. 2016. Dialog-based language learning. In *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 829–837. http://papers.nips.cc/paper/6264-dialog-based-language-learning.pdf.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. Association for Computational Linguistics, pages 404–413. http://www.aclweb.org/anthology/W13-4065.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, IJCAI'16, pages 2308–2314. http://dl.acm.org/citation.cfm?id=3060832.3060944.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada. https://arxiv.org/pdf/1704.01696.pdf.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179. http://ieeexplore.ieee.org/document/6407655/.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*. AAAI Press/MIT Press, Portland, OR, pages 1050–1055. http://www.cs.utexas.edu/users/ai-lab/?zelle:aaai96.