A Goal-Driven Natural Language Interface for Creating Application Integration Workflows

Michelle Brachman[1], Christopher Bygrave[2], Tathagata Chakraborti[1], Arunima Chaudhary[1], Zhining Ding[3], Casey Dugan[1], David Gros[4], Thomas Gschwind[1], James Johnson[1], Jim Laredo[1], Christoph Miksovic[1], Qian Pan[1], Priyanshu Rai[5], Ramkumar Ramalingam[2], Paolo Scotton[1], Nagarjuna Surabathina[2], Kartik Talamadupula[1]

[1]IBM Research [2]IBM Cloud & Cognitive Software [3]Wellesley College [4]UC Davis [5]Persistent Systems

Abstract

Web applications and services are increasingly important in a distributed internet filled with diverse cloud services and applications, each of which enable the completion of narrowly defined tasks. Given the explosion in the scale and diversity of such services, their composition and integration for achieving complex user goals remains a challenging task for end-users and requires a lot of development effort when specified by hand. We present a demonstration of the *Goal Oriented Flow Assistant* (GOFA) system, which provides a natural language solution to generate workflows for application integration. Our tool is built on a three-step pipeline: it first uses Abstract Meaning Representation (AMR) to parse utterances; it then uses a knowledge graph to validate candidates; and finally uses an AI planner to compose the candidate flow. We provide a video demonstration of the deployed system as part of our submission.

Introduction

In the cloud and application driven internet of today, the composition and integration of various application constituted workflows is an important problem. Many end-user tasks cannot be accomplished without composing (*connecting*) different applications together in an *integration flow*. This task – akin to web service composition (Srivastava and Koehler 2003; Hoffmann, Bertoli, and Pistore 2007) – is a common development task for programmers and non-programmers, and can benefit from automation (Gschwind 2002). Recently, enterprise industrial tools like App Connect [1], Zapier [2], and Microsoft Power Automate [3] have begun to provide visual and form-based end-user programming environments for application integration. Even in such end-user programming environments, users still face significant barriers (Ko, Myers, and Aung 2004): (i) the significant scale and diversity of the number of applications renders *discovery* a tedious and oftentimes impossible task; and (ii) the variety and complexity of the inputs and outputs of various applications introduces additional barriers to handcrafted *composition* by users.

[1]https://www.ibm.com/cloud/app-connect

[2]https://zapier.com

[3]https://powerautomate.microsoft.com

Natural language has the potential to reduce barriers and learning curves for complex tasks (Linder et al. 2013; Sales, Handschuh, and Freitas 2017), like end-user programming for web service and application composition. Even for users with sufficient programming experience, natural language generation of service composition tasks could increase the efficiency of the interaction and composition by reducing GUI interactions and by providing automatically generated flow candidates. In this paper, we introduce the *Goal-Oriented Flow Assist* (GOFA) system – a system that generates web application integration flows from natural language in the context of App Connect. GOFA automatically produces application integration flows from a user utterance specified as a natural language sentence. GOFA uses a three step pipeline: 1) utterance parsing with AMR, 2) validation of candidates with Knowledge Graph, and 3) composition with an AI planner.

Background

Our system has been implemented within App Connect. App Connect is a GUI-based tool that enables users to create application integration *flows*. In these flows, users can connect sequences of applications without having to learn or to fully understand the APIs (Application Programming Interfaces) underlying each application. A *connector* in App Connect is a visual component that encapsulates connectivity to an application; manipulates any data received; and optionally transfers its output to another downstream connector. A connector can react to a notification; read data; or modify data by creating, updating or deleting a record in an application. Some connectors may offer more than one record type or *business object* that can be manipulated.

For example, a user can create a flow that does the following: when there is a new attendee on Eventbrite, create a contact in HubSpot and add them to a subscription list on MailChimp. We focus on trigger-action flows, which include one trigger event and one or more actions that follow a trigger. In this case the trigger is a notification from Eventbrite that would transfer the new attendee as a contact to the Hubspot application and create a subscription for that attendee in Mailchimp (for future notifications). We demonstrate a tool that can enable users to initiate the creation of this type of application integration flow with natural language.

Goal Oriented Flow Assist (GOFA)

The architecture of our system is shown in Figure 1. First, the Abstract Meaning Representation (AMR) service parses the sentence into a structured representation. Next, the Integration Knowledge Graph (IKG) matches the parts of this structured representation against concepts provided by App Connect. Finally, the AI Planner uses all this information to compose the final workflow.

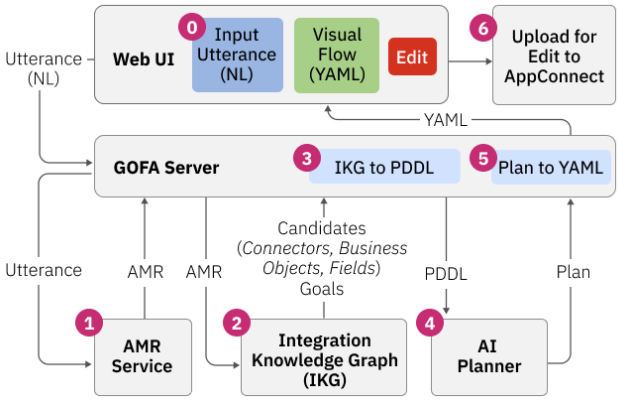

Figure 1: Architecture of the GOFA System.

Abstract Meaning Representation (AMR)

First, we use AMR (Banarescu et al. 2013; Naseem et al. 2019; Astudillo et al. 2020) to abstract away from syntactic idiosyncrasies in the utterance and generate its representation as rooted, directed, edge-labeled, leaf-labeled graphs that are easy to traverse. We then use this graph to resolve the utterance into tasks using parts-of-speech and edge types associating them. In our case, a task is primarily composed of: a connector, the business object associated with that connector, and the operation (action) that needs to be performed on that business object. Once the task elements have been resolved, we further use heuristics obtained from the graph to establish this task as a trigger or a simple action that may follow a trigger found in the set of all the tasks resolved from AMR. The tasks obtained as a result of this exercise become input for the IKG to find best matches.

Integration Knowledge Graph (IKG)

The IKG analyzes the metadata of the connectors available in App Connect and identifies common items between different connectors representing them in a graph structure. This information contains a description of applications, business objects those applications can handle, and operations allowed on those business objects.

The IKG is then enriched by discovering links between similar entities, also referred to as latent links. For instance, both DropBox and Microsoft OneDrive provide file objects even if they may be named differently within their respective APIs. Latent link discovery is done using embedding distance (Devlin et al. 2019) and Graph Neural Networks (Sheikh et al. 2020) to capture the relationship of a given entity to its surroundings in the graph. The IKG thus reflects edges connecting applications through common business objects and, therefore, the set of feasible flows. An augmented list of candidates comes out of the IKG and on to the next step in the GOFA pipeline. Additionally, after the relevant parts have been identified in the previous step, the word fragments are passed to the IKG to return the closest nodes along with their neighborhood in the IKG, and an associated relevance score. This information is subsequently consumed by the planner.

AI Planner

To complete the flow composition process, we apply AI planning techniques to generate one or more flows depending on the candidates provided by the IKG in the previous step. The AI Planner that we use – *Masterplan* – is based off of prior work in the automated planning community (Katz 2018; Katz et al. 2018a,b; Katz, Sohrabi, and Udrea 2020; Katz and Sohrabi 2020). The planner receives the candidates identified via the AMR and IKG steps; additionally, it also receives the action schemas for each of the idenfitied candidates. These schemas consist of the name of the action (operation), as well as the *ins* and *outs* of each action. This structure is very similar to web services, which have inputs and outputs that need to be matched up to produce valid compositions. Similar to this, our AI planner utilizes all the information provided to sequence the candidates into a valid candidate flow. These proposed flows are then translated into the native YAML format that is understood by App Connect to render and allow further editing of the proposed flow.

User Interface

The final step in our system pipeline visualizes the flow candidate from the previous step as part of a visual interface. To use this interface, users currently type utterances into a textbox and click submit to generate their flow. There are example utterances provided that users can reference. For demonstration and debugging purposes, we currently also provide a visualization of the AMR parsing and IKG mapping – these can be turned off in order to improve the return time of the system. Future work includes providing explanations and opportunities to repair generated flows. Additionally, further work on human-AI collaboration will be important to bring the user's mental model closer to the system capabilities, especially considering the three-part pipeline for utterance-to-flow generation.

Demonstration

Our demonstration showcases all the components of the GOFA pipeline: starting from the user input of the utterance, all the way to the visualization of the candidate flow. We utilize additional visualizations available in our current interface as a way to dive deeper into the details of the AMR (of interest to an NLP audience) and the IKG (of interest to semantic web and graph representations audiences). Our system – which is already deployed online – will be made available so that conference participants can interact with the system on their own; and test different varieties of inputs.

References

Astudillo, R. F.; Ballesteros, M.; Naseem, T.; Blodgett, A.; and Florian, R. 2020. Transition-based Parsing with Stack-Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 1001–1007.

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, 178–186.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Gschwind, T. 2002. *Adaptation and Composition Techniques for Component-Based Software Engineering*. Ph.D. thesis, Technische Universität Wien.

Hoffmann, J.; Bertoli, P.; and Pistore, M. 2007. Web service composition as planning, revisited: In between background theories and initial state uncertainty. In *AAAI*, volume 7, 1013–1018.

Katz, M. 2018. Cerberus: Red-black heuristic for planning tasks with conditional effects meets novelty heuristic and enchanced mutex detection. *Ninth International Planning Competition (IPC-9): planner abstracts*, 47–51.

Katz, M.; and Sohrabi, S. 2020. Reshaping diverse planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34.06, 9892–9899.

Katz, M.; Sohrabi, S.; Samulowitz, H.; and Sievers, S. 2018a. Delfi: Online planner selection for cost-optimal planning. *IPC-9 planner abstracts*, 57–64.

Katz, M.; Sohrabi, S.; and Udrea, O. 2020. Bounding Quality in Diverse Planning. *HSDIP 2020*, 49.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018b. A novel iterative approach to top-k planning. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Ko, A. J.; Myers, B. A.; and Aung, H. H. 2004. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*, 199–206. IEEE.

Linder, J.; Laput, G.; Dontcheva, M.; Wilensky, G.; Chang, W.; Agarwala, A.; and Adar, E. 2013. PixelTone: A multimodal interface for image editing. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 2829–2830.

Naseem, T.; Shah, A.; Wan, H.; Florian, R.; Roukos, S.; and Ballesteros, M. 2019. Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4586–4592. Florence, Italy: Association for Computational Linguistics.

Sales, J. E.; Handschuh, S.; and Freitas, A. 2017. Semeval-2017 task 11: end-user development using natural language. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 556–564.

Sheikh, N.; Qin, X.; Reinwald, B.; Miksovic, C.; Gschwind, T.; and Scotton, P. 2020. Knowledge Graph Embedding using Graph Convolutional Networks with Relation-Aware Attention. In *The 2nd International Workshop on Deep Learning on Graphs: Methods and Applications (KDD-DLG 2020)*.

Srivastava, B.; and Koehler, J. 2003. Web service composition-current solutions and open problems. In *ICAPS 2003 workshop on Planning for Web Services*, volume 35, 28–35.