A Human-in-the-loop API Sequencing Tool Powered by AI Planning

Jungkoo Kang · Tathagata Chakraborti · Junkyu Lee · Michael Katz · Shirin Sohrabi
Gaodan Fang · Punleuk Oum · Prabhat Reddy · Diego del Rio · Debashish Saha · Gegi Thomas
Benjamin Herta · Jim Laredo · Alina Rotarescu · Alice Zhang · Suzette Samoojh

IBM

Abstract

In this demo, we present a real-time API recommender system powered by an automated planner. Our tool generates multiple API recommendations for over 600 APIs within a bound of 5 seconds. By inputting a partial list of APIs, our tool dynamically fills in missing components to create a more functional or complete workflow. While this task has been historically perceived as solely a data-driven endeavor, we demonstrate how a planner can be harnessed to utilize both association information and structural dependencies between different APIs. As part of the demonstration, we also report on data gathered from the initial deployment of the tool.

Introduction

Lowering the effort and expertise needed for workflow construction is an area of active research for automation tools (Chakraborti et al. 2022). In the past, we have shown how declarative specifications can assist a user in constructing sophisticated workflows using a planner, in the context of goal-oriented conversational agents (Muise et al. 2019; Sreedharan et al. 2020; Chakraborti et al. 2021) and through the composition of units of automation in the form of APIs and "skills" (Chakraborti et al. 2020). Specifically for API-based automation, our efforts have largely been confined to natural language interactions (Brachman et al. 2022, 2023) that serve as a jump-off point to a more sophisticated build experience in a graphical interface. In this demo, we focus on this latter build experience where the user is in the process of adding to a workflow under construction with the help of recommendations from the system.

The API Recommendation Task

A series of interconnected APIs can form a comprehensive workflow to accomplish complex tasks like processing job applications, handling event registrations, or managing file sets, among other high-level objectives. The API recommendation task involves recommending, given an existing sequence of APIs, new APIs that can be added to the sequence. Through multiple iterations of recommendations and adoption, a user can build out from scratch new workflows that automate their own processes. Unlike the plan completion task (Tian, Zhuo, and Kambhampati 2016), here the task is decidedly human-in-the-loop and not merely a sequence completion task. During the recommendation process, it is essential to offer multiple alternative approaches for completing a workflow. Each option should represent distinct high-level tasks while allowing users the flexibility to either accept or reject these suggested recommendations. The recommended APIs can come in two forms:

PBA or **P**revious **B**est **A**ction refers to the recommendation of APIs *before* a given position in a sequence of APIs. In this demo, we focus exclusively on the PBA task.

NBA or **N**ext **B**est **A**ction refers to the recommendation of APIs *at the end* of a sequence of APIs.

Related Work The API recommendation problem has parallels to web service composition (Lemos, Daniel, and Benatallah 2015), where planning-based approaches have found a natural home (Srivastava and Koehler 2003). This includes modeling of ontological knowledge (Hoffmann et al. 2007; Sirbu and Hoffmann 2008) for the planner as well as compiling user preferences (Sohrabi, Prokoshyna, and McIlraith 2006) into the reasoning problem up front. On the other hand, the tool being presented serves as a recommender system, thereby requiring the ability of the planner to come up with multiple alternative suggestions per request, as well as operate in a human-in-the-loop setting where new user inputs are folded in interactively – these considerations, along with how we compile similarity and historical data into the planning task, makes this a unique deployment.

Modeling Considerations for the Planner

While computing what can be a good recommendation for a PBA task, the following considerations come to mind:

1. Whether this single API or a series of new ones can generate inputs for subsequent APIs within an existing sequence. Notably, if multiple new APIs are introduced, all those in the candidate prefix sequence should be considered as potential recommendation candidates;
2. Whether the API, by its description or name, is relevant to the topic of the other APIs in the sequence; and
3. Whether the user has historically executed that API in that context in the past.

The first criterion is a planning task: computing the least sized sequence that can satisfy all the data requirements (inputs to the subsequent APIs) of a flow. While typically the

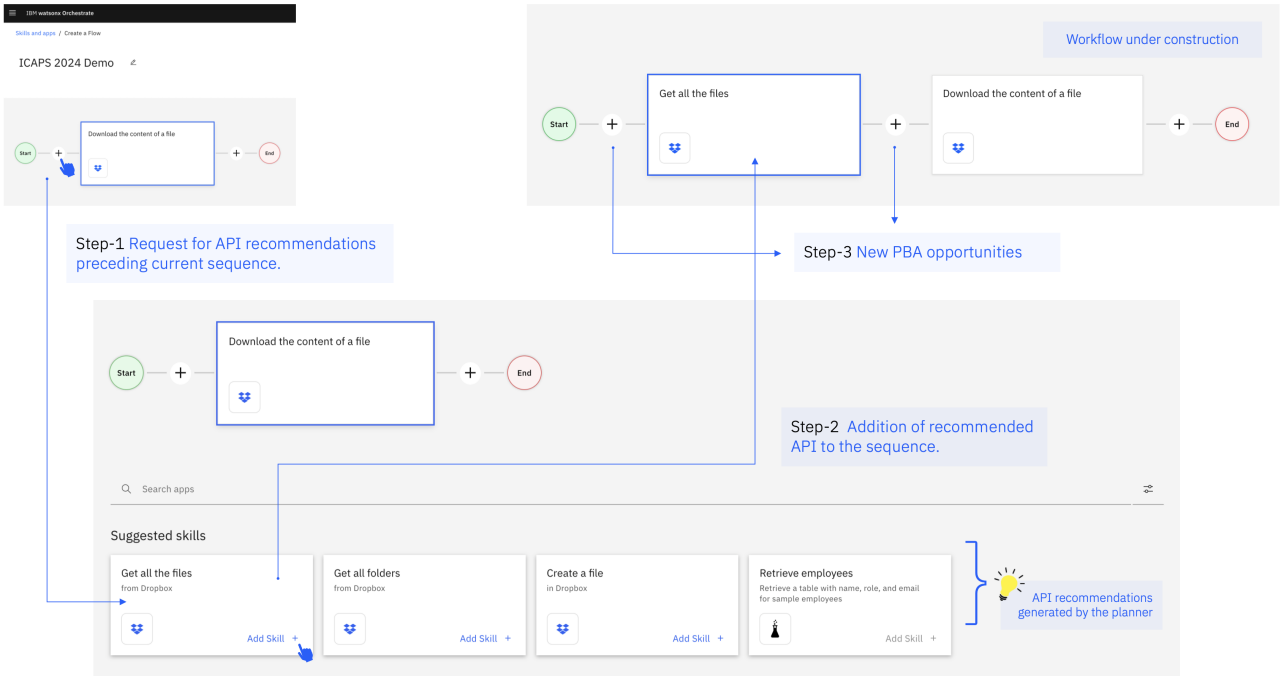

Figure 1: A user is building a workflow involving file operations using the Dropbox API. The planner in the tool assists the user by suggesting one or more APIs to add to the workflow under construction. Video: https://youtu.be/o75mNgXIwSs.

(API) recommendation task has been approached purely as a learning task, a planner gives us two potential advantages. First, compilation of association probability and usage data into the cost model allows us to model both data-driven characteristics as well as those that require logical reasoning e.g. piping information across components. While purely learning approaches can do the former, they are typically quite poor at the latter. Second, since not all the considerations are dependent on past usage data or knowledge acquired from generic data, we are not fully hamstrung by the cold start problem of traditional recommender systems.

Based on the mentioned factors, a PDDL (Planning Domain Definition Language) specification is dynamically created from the input data using `tarski` (Francés, Ramirez, and Collaborators 2018). Subsequently, various sequences are generated through the K* planner (Lee, Katz, and Sohrabi 2023), preserving orders only between the actions that are associated with APIs (Katz et al. 2024). The input data includes Open API Specs as well as a collection of associated APIs, represented by pairings of API names. These associated APIs were derived through two methods: from user-defined sequences or from semantically similar APIs based on the API name, path, and the operation (see https://swagger.io/docs/specification/paths-and-operations). These were identified by a compact (∼100Mb) sentence transformer model. Following this, the generated plans are further processed to offer multiple suggestions for the desired position within the developing workflow.

Example Interaction Figure 1 shows a sample interaction in the tool. The user has begun constructing a workflow with the Dropbox (download file) API. The PBA request produces several possible APIs to appear before it, such as fetching files or creating one – which are all within the space of Dropbox APIs that can precede the download operation already specified – or involving different APIs that may be part of known business processes (such as hiring) usually associated with the usage of the already selected API. The user selects the one they want (or use the search bar to discover other APIs if their desired API is not among the recommendations) and continue completing their workflow.

Measurement	Initial Deployment
Recommendations per day	800+ per day
Percentage PBA (versus NBA) request	33%
Percentage adoption of recommendation	50%
Time taken to produce recommendations	2-3 seconds
Length of sequences built	2 (majority) - 6

Table 1: Preliminary usage data from first deployment.

PBA in Action Table 1 presents some preliminary data gathered during the first months of use of the tool. A 33% rate for PBA requests demonstrates the importance of the PBA use case (and points to how users think about the sequencing task as either goal-directed or backward from the goal). A 50% hit rate is promising but also indicates potential room for improvement.

Demonstration Logistics During the demo session, the audience will be able to play with the tool, create their own workflows, and explore the generated suggestions. They will also be able to investigate the modeling nuances in the PDDL specifications generated by the tool.

References

Brachman, M.; et al. 2022. A Goal-driven Natural Language Interface for Creating Application Integration Workflows. In *AAAI Demo Track*.

Brachman, M.; et al. 2023. Follow the Successful Herd: Towards Explanations for Improved Use and Mental Models of Natural Language Systems. In *IUI*.

Chakraborti, T.; Agarwal, S.; Brimijoin, K.; Agarwal, P.; Rizk, Y.; Moran, D. S.; Boag, S.; and Khazaeni, Y. 2021. Planning for Automated Composition of Aggregated Assistants. In *ICAPS Demonstration*.

Chakraborti, T.; Agarwal, S.; Khazaeni, Y.; Rizk, Y.; and Isahagian, V. 2020. D3BA: A Tool for Optimizing Business Processes Using Non-Deterministic Planning. In *BPM Workshop on AI4BPM*.

Chakraborti, T.; Rizk, Y.; Isahagian, V.; Aksar, B.; and Fuggitti, F. 2022. From Natural Language to Workflows: Towards Emergent Intelligence in Robotic Process Automation. In *BPM*.

Francés, G.; Ramirez, M.; and Collaborators. 2018. Tarski: An AI Planning Modeling Framework. https://github.com/aig-upf/tarski.

Hoffmann, J.; Bertoli, P.; Pistore, M.; et al. 2007. Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty. In *AAAI*.

Katz, M.; Lee, J.; Kang, J.; and Sohrabi, S. 2024. Some Orders Are Important: Partially Preserving Orders in Top-Quality Planning. In *SoCS*.

Lee, J.; Katz, M.; and Sohrabi, S. 2023. On K* Search for Top-k Planning. In *SoCS 2023*.

Lemos, A. L.; Daniel, F.; and Benatallah, B. 2015. Web Service Composition: A Survey of Techniques and Tools. *ACM Computing Surveys (CSUR)*.

Muise, C.; Chakraborti, T.; Agarwal, S.; Bajgar, O.; Chaudhary, A.; Lastras-Montano, L. A.; Ondrej, J.; Vodolan, M.; and Wiecha, C. 2019. Planning for Goal-Oriented Dialogue Systems. *arXiv:1910.08137*.

Sirbu, A.; and Hoffmann, J. 2008. Towards Scalable Web Service Composition with Partial Matches. In *IEEE International Conference on Web Services*.

Sohrabi, S.; Prokoshyna, N.; and McIlraith, S. A. 2006. Web Service Composition via Generic Procedures and Customizing User Preferences. In *International Semantic Web Conference*.

Sreedharan, S.; Chakraborti, T.; Muise, C.; Khazaeni, Y.; and Kambhampati, S. 2020. D3WA+ – A Case Study of XAIP in a Model Acquisition Task for Dialogue Planning. In *ICAPS*.

Srivastava, B.; and Koehler, J. 2003. Web Service Composition – Current Solutions and Open Problems. In *ICAPS Workshop on Planning for Web Services*.

Tian, X.; Zhuo, H. H.; and Kambhampati, S. 2016. Discovering Underlying Plans Based on Distributed Representations of Actions. *AAMAS*.