Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)
Demonstrations Track

Towards a Resilient Intelligent Automation System

Segev Shlomov , Sami Marreed , Avi Yaeli
IBM Research
segev.shlomov1@ibm.com, sami.marreed@ibm.com, aviy@il.ibm.com

Abstract

Intelligent Process Automation (IPA) solutions must adapt to changes in user interfaces autonomously, without manual intervention. Addressing this critical challenge and aiming to advance the state-of-the-art, last year we introduced the IPA Challenge competition at IJCAI 2023. This demo paper presents IDA - our novel UI automation solution, developed to tackle complex resiliency issues. Leveraging the capabilities of large language models and employing grounded instructions, our system demonstrates a significant advancement towards resilient IPA. We provide an overview of the IPA Challenge, detail the architecture of our system, and illustrate its effectiveness in overcoming the resiliency challenges. A link to the demo video can be found at: https://youtu.be/G5nI3V9Umjc

1 Introduction

Web Automation primarily involves the automatic control of web browsers to perform repetitive tasks such as data entry, content scraping, and interaction with web applications [Sugiura and Koseki, 1998; Little *et al.*, 2007; Leshed *et al.*, 2008; Le and Gulwani, 2014]. Early web automation tools were primarily developer-centric, requiring a good understanding of web technologies such as HTML, CSS, and JavaScript.

More recently, UI Automation has been evolving into an important solution for non-technical users overwhelmed with repetitive and ad-hoc tasks across enterprise digital platforms. Beyond just streamlining existing work, UI Automation plays a key role in unlocking the value of digital workers and transforming how businesses operate. Given recent advancements in Large Language Models (LLMs), could they potentially overcome the technical challenges and simplify no-code tools to better suit the skills of non-technical users?

Despite these recent advancements, commercial Intelligent Process Automation (IPA) solutions, including UIPAth [Tripathi, 2018], Automation Anywhere [Automation Anywhere, 2023], Microsoft Power Automate [Guilmette, 2020], and IBM RPA [IBM, 2022], encounter challenges in achieving autonomous resilience, particularly in adapting to dynamic UI implementations or layouts. While recent generative AI-based assistants, such as ChatGPT [OpenAI, 2024] and Adept [Fuyu-Heavy, 2024], and LMM-based web agents [Zheng *et al.*, 2024a; Wang *et al.*, 2024; He *et al.*, 2024; Koh *et al.*, 2024], offer promising approaches by dynamically adjusting their automation strategies in response to UI changes, they often fall short in accurately grounding their actions for effective UI interaction [Zheng *et al.*, 2024b]. These fundamental issues prevent the adoption of automation solutions by enterprises, limiting the fulfillment of a broader range of user needs [Schwartz *et al.*, 2023].

1.1 The IPA Challenge Competition

The IPA Challenge Competition, jointly organized by industry and academia at IJCAI 2023, focused on enhancing the resiliency of IPA through AI. It required participants to develop solutions for automating UI tasks adaptable to changing conditions, a crucial aspect of robust IPA systems. Details of the competition can be found at http://iparesiliency-ijcai23.github.io/. Participants were tasked with creating an AI bot to execute four progressively challenging automation tasks, based on user data from an Excel spreadsheet

Task 1 - Beginner involved form-filling with data that changes position and labels (e.g., the input "Role in company" might be changed to "Job description") after each submission, testing the bot's semantic understanding of form fields. *Task 2 - Advanced* extended Task 1's complexity with variably positioned form labels and changes in the DOM architecture of the HTML (e.g., the label of the input "Manager" will be changed to "Supervisor" and will become a placeholder), challenging the bot to navigate structure changes. *Task 3 - Pro* required user lookup and form-filling for "Add" or "Remove" actions, with form elements and labels changing randomly, necessitating advanced adaptation techniques. *Task 4 - Pro Max* extended Task 3's complexity by introducing a "Reason" field with dynamically changing values and implementations.

Overall, eight teams participated in the challenge utilizing a mix of established IPA tools and custom AI technologies. Only two teams could navigate through the beginner and advanced tasks. The Pro, and Pro-max tasks, requiring nuanced UI understanding and element disambiguation, remained uncompleted by all, underscoring the existing gaps in IPA capabilities. Consequently, the challenge has been opened to the public, aiming to serve as a resilience benchmark and spur further innovation in the field.

2 IDA - A Resilient IPA System

We introduce IDA (Intelligent Digital Apprentice) - an innovative no-code UI automation system for business users that successfully passes all the IPA challenges. While IDA includes many features to enable business users to teach it to perform UI automation, in this demo paper we focus mostly on the runtime components that enable IDA to be resilient to changes in the UI. IDA can perform automation flows by following instructions in natural language (Figure 1). Every instruction includes an action type, reference name of the UI element, and optional value parameter. IDA can be triggered to perform the automation on each row of an Excel file and leverage data in Excel as input parameters to the automation flow. In the IPA challenge each row corresponds to a user that needs to be added or updated in a system, and IDA needs to dynamically map column headers of the Excel to UI input elements which constantly change their names and underlying implementation. In the following subsections we describe the key algorithms that enable IDA to execute automation flows in a robust and resilient manner.

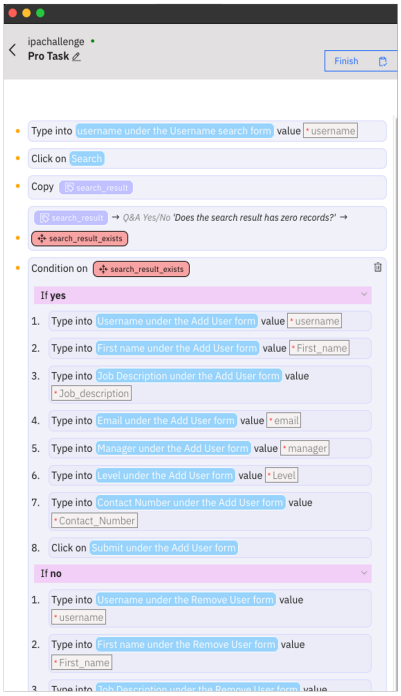

Figure 1: The instructions flow. The fourth step from above is the semantic state UI understanding and the fifth step is the condition based on the result of the state.

2.1 Grounded Element Selection

To automate natural language-based instructions, IDA must understand the UI semantically. For every UI element, IDA needs to understand its type and name (based on visible text). IDA employs several LLM-based pipelines which enables it to ground the instructions to semantic elements that exist in the application. The main steps of the algorithms are:

Step 1 - Discovering new heuristics for detecting UI elements. IDA uses heuristic rules to detect UI element types on the screen. For robustness of the solution, IDA uses LLMs to keep learning about new UI frameworks and implementations and keep its heuristics up to date for complete UI detection coverage. Given a snapshot of HTML from the DOM, the system first prompts an LLM with an HTML snippet of the current UI, and asks it to suggest several HTML heuristics in order to best collect UI elements. This includes identifying buttons, links, inputs, and semantic states of UI elements. The output of this step is a JSON file with the heuristics which is later will be used to extract all the elements. As the HTML DOM is super complex, the motivation behind this step is to use the power and knowledge of LLMs in HTML and web framework specifications.

Figure 2: Heuristics construction for elements collection. The prompt is enriched with few-shot examples based on common element collection heuristics

Step 2 - Semantic UI understanding. In this step, the system applies the updated HTML heuristics to the current page and identifies all the candidate UI elements and their corresponding names. IDA uses CSS selectors to query the DOM (and the Shadow-DOM) based on expanded and refined HTML specification heuristics presented in Figure 2. To determine naming, it combines elements with their near-by-labels using reference attributes and the element's positions. This is done using both pixel-wise proximity algorithm and DOM-based distance. Finlay, the system constructs elements hierarchy such as lists, groups, and forms. For example, our system knows that some inputs are part of the "Add form" while the other belongs to the "Remove form". The output of this step is a full textual (and visual) representation of the current browser content with a unique name for each element.

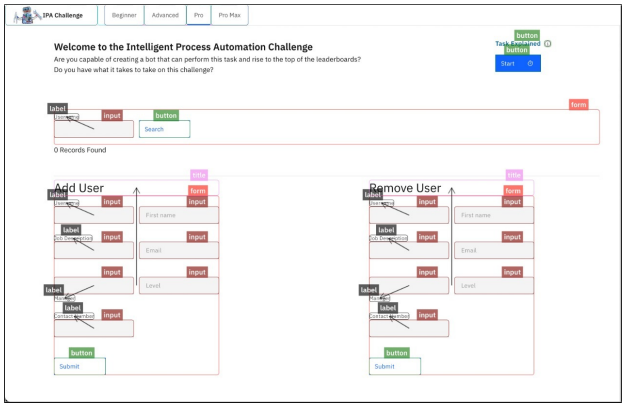

Figure 3: Semantic page understanding with hierarchy visual output

Step 3 - Instruction to UI element grounding. In this step, IDA performs UI element grounding by matching the referenced element in the instruction to the list of UI elements performed by the Semantic UI Understanding step. IDA first filters all the UI elements that are not relevant to the instruction by using a two-step hierarchy semantic similarity. That is, it compares the cosine similarity score between the embeddings on both the group/form level and then on the elements level itself. In real-world web pages, when the number of elements on the screen can be huge, this is a super crucial step. Next, it prompts an LLM to match the instruction to the list of elements. The LLM task is to choose the right element ID to perform the action. Once grounded, the system then executes the action on the chosen UI element. The usage of multiple-step semantic meaning and an LLM approach enables the system to perform robustly.

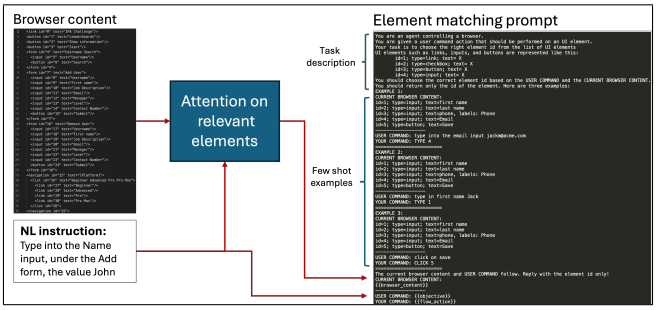

Figure 4: Grounding Element via LLM following a multiple stage semantic similarity

2.2 Detection of Semantic Element State

In some cases, detecting the element type and name is not enough and IDA must detect complex UI elements and their internal states. For example in more complex semantic UI elements such as search results tables or attachments, IDA employs an innovative approach to recognize the elements, their names, and their state at a specific state within the automation flow. IDA includes a definition of possible semantic objects and states (e.g., a search result has “one record” or “multiple records” state, whereas file attachment may be “present” or “absent”) and uses an LLM to parse HTML snippets and detect the semantic objects and their current states. The LLM task is also used to generate a user-friendly name for the semantic object, as well as to generate a JavaScript code that can evaluate the state of the object at runtime. As presented in the following task prompt (Figure 5) and response from the LLM (Figure 6), the LLM is able to detect the semantic element type (table), its user-friendly name (“Companies table”), and its semantic state (“has multiple records”).

In the context of full automation flow, IDA has the ability to perform semantic action on these UI elements and apply a condition on the answer, resulting in different behavior of the system. Currently, this feature was not tested for resiliency as the CSS selector of “0 records found” was not changed in the challenge. Future work will have to tackle these types of issues.

```
As a helpful assistant, Your task is for a given HTML Element to identify its functional name, assign its
    object type and semantic state, and, for the assigned semantic state, generate a javascript code that
    evaluates the state.

Semantic states knowledge:
    type: file.
    states: file exists, file doesn't exist Object.

    type: table.
    states: table has one record, table has multiple records, table has no records.
Input HTML:
    <table><tbody><tr><th>Company</th><th>Contact</th><th>Country</th></tr><tr><td>Ernst Handel</td><td>
    Roland Mendel</td><td>Austria</td></tr><tr><td>Island Trading</td><td>Helen Bennett</td><td>UK</td></
    tr><tr><td>Laughing Bacchus Winecellars</td><td>Yoshi Tannamuri</td><td>Canada</td></tr><tr><td>
    Magazzini Alimentari Riuniti</td><td>Giovanni Rovelli</td><td>Italy</td></tr></tbody></table>
Answer:
    Functional name: Companies table
    Matching object type: table
    Semantic state: table has multiple records.
    Explanation: the current table HTML has 4 rows, meaning that the matching semantic state is that the
    table has multiple records.
Javascript expression:
    ```
    function checkIfTableHasMultipleRecords() {
    try {
    return document.querySelector('table tbody').querySelectorAll('tr').length > 1; }
    catch (error) {
     // In case of an error (e.g., table or tbody not found),
      return false return false;
     }
    }
    ```

Input HTML:
{{INPUT}}
```

Figure 5: Detection of Semantic Element State - 1-shot example: An HTML element constitutes a task input. The task output consists of textual information on the HTML element and a Javascript function.

```
Semantic states knowledge:
    type: file.
    states: file exists, file doesn't exist Object.

Input HTML:
    <div class="oxd-grid-item oxd-grid-item--gutters" data-v-c93bdbf3="" data-v-0da39c26=""><!----><div
    class="orangehrm-file-input" data-v-2d6a04e6=""><div class="oxd-input-group oxd-input-field-bottom-
    space" data-v-957b4417="" data-v-2d6a04e6=""><div class="oxd-input-group__label-wrapper" data-v-957
    b4417=""><!----><label class="oxd-label" data-v-30ff22b1="" data-v-957b4417="">Resume</label></div><
    div class="" data-v-957b4417=""><input type="file" class="oxd-file-input" data-v-1b0f05a0=""><div
    class="oxd-file-div oxd-file-div--active" data-v-1b0f05a0=""><div class="oxd-file-button" data-v-1
    b0f05a0="">Browse</div><div class="oxd-file-input-div" data-v-1b0f05a0="">No file selected</div><i
    class="oxd-icon bi-upload oxd-file-input-icon" data-v-bddebfba="" data-v-1b0f05a0=""></i></div></div
    ><!----></div><p class="oxd-text oxd-text--p orangehrm-input-hint" data-v-7b563373="" data-v-2d6a04e6
    ="">Accepts .docx, .doc, .odt, .pdf, .rtf, .txt up to 1MB</p></div></div>
Answer:
    Functional name: File upload input
    Matching object type: file
    Semantic state: file doesn't exist.
    Explanation: The current HTML code represents a file upload input, and the semantic state is that no
    file is currently selected.
Javascript expression:
    ```
    function checkIfFileDoesntExist() {
    try {
    return document.querySelector('.oxd-file-input-div').innerText === 'No file selected'; }
    catch (error) {
     // In case of an error (e.g., file input not found),
      return false; }
    }
```

Figure 6: Detection of Semantic Element State - LLM response for a File input UI element: The LLM determines the semantic state of “file doesn’t exist” and the code required to compute this state

# 3 Discussion

IDA stands out from state-of-the-art systems through its integration of LLMs for robust semantic understanding and dynamic element matching via grounding instructions to semantic UI elements. This enables IDA to handle complex resiliency issues and surpass competitors in the IPA Challenge. Beyond the IPA challenge, we have put IDA to the test in a wide range of additional real-world applications across diverse platforms such as SAP and Salesforce, where IDA demonstrated exceptional generalization without the need for extensive reconfiguration. This pivotal capability may mark a new era towards a more resilient and trustworthy IPA as well as set new adaptability benchmarks. Despite its innovative approach, IDA faces limitations in scenarios with complex or ambiguous UI elements, where reliance on semantic similarity may introduce inaccuracies. This points to the need for further research to improve semantic analysis and element matching. The IPA Challenge, serving as an open benchmark, underscores the importance of resilient automation solutions. IDA’s performance encourages the pursuit of new benchmarks in the field, driving innovation and research toward more adaptive and robust IPA solutions.
```

References

[Automation Anywhere, 2023] Automation Anywhere. www.automationanywhere.com, 2023.

[Fuyu-Heavy, 2024] Adept. Fuyu-Heavy. https://www.adept.ai/, 2024.

[Guilmette, 2020] Aaron Guilmette. *Workflow Automation with Microsoft Power Automate: Achieve digital transformation through business automation with minimal coding*. Packt Publishing Ltd, 2020.

[He *et al.*, 2024] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*, 2024.

[IBM, 2022] IBM. Ibm robotic process automation. www.ibm.com/products/robotic-process-automation, 2022.

[Koh *et al.*, 2024] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

[Le and Gulwani, 2014] Vu Le and Sumit Gulwani. Flashextract: A framework for data extraction by examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 542–553, 2014.

[Leshed *et al.*, 2008] Gilly Leshed, Eben M Haber, Tara Matthews, and Tessa Lau. Coscripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1719–1728, 2008.

[Little *et al.*, 2007] Greg Little, Tessa A Lau, Allen Cypher, James Lin, Eben M Haber, and Eser Kandogan. Koala: capture, share, automate, personalize business processes on the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 943–946, 2007.

[OpenAI, 2024] OpenAI. (2024). ChatGPT (3.5) OpenAI. https://chat.openai.com, 2024.

[Schwartz *et al.*, 2023] Sivan Schwartz, Avi Yaeli, and Segev Shlomov. Enhancing trust in llm-based ai automation agents: New considerations and future challenges, 2023.

[Sugiura and Koseki, 1998] Atsushi Sugiura and Yoshiyuki Koseki. Internet scrapbook: automating web browsing tasks by demonstration. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 9–18, 1998.

[Tripathi, 2018] Alok Mani Tripathi. *Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool–UiPath*. Packt Publishing Ltd, 2018.

[Wang *et al.*, 2024] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024.

[Zheng *et al.*, 2024a] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

[Zheng *et al.*, 2024b] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.