SmartFlow: Robotic Process Automation using LLMs

Arushi Jain, Shubham Paliwal, Monika Sharma, Lovekesh Vig, Gautam Shroff
TCS Research
New Delhi, India
(j.arushi,shubham.p3,monika.sharma1,lovekesh.vig,gautam.shroff)@tcs.com

ABSTRACT

Robotic Process Automation (RPA) systems face challenges in handling complex processes and diverse screen layouts that require advanced human-like decision-making capabilities. These systems typically rely on pixel-level encoding through drag-and-drop or automation frameworks such as selenium to create navigation workflows, rather than visual understanding of screen elements. In this context, we present SmartFlow, an AI-based RPA system that uses large language models (LLMs) coupled with deep-learning based image understanding. Our system can adapt to new scenarios, including changes in the user interface and variations in input data, without the need for human intervention. SmartFlow uses computer vision and natural language processing to perceive visible elements on the graphical user interface (GUI) and convert them into a textual representation. This information is then utilized by LLMs to generate a sequence of actions that are executed by a scripting engine to complete an assigned task. To assess the effectiveness of SmartFlow, we have developed a dataset that includes a set of generic enterprise applications with diverse layouts, which we are releasing for research use. Our evaluations on this dataset demonstrate that SmartFlow exhibits robustness across different layouts and applications. SmartFlow can automate a wide range of business processes such as form filling, customer service, invoice processing, and back-office operations. SmartFlow can thus assist organizations in enhancing productivity by automating an even larger fraction of screen-based workflows. The demo-video, supplementary material and dataset are available at https://smartflow-4c5a0a.webflow.io/.

CCS CONCEPTS

• **Human-centered computing** → *Accessibility systems and tools.*

KEYWORDS

Robotic Process Automation, Large Language Models, Intelligent Automation, Computer Vision

ACM Reference Format:
Arushi Jain, Shubham Paliwal, Monika Sharma, Lovekesh Vig, Gautam Shroff. 2023. SmartFlow: Robotic Process Automation using LLMs. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (CIKM'23).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

CIKM'23, Oct 21–25, 2023, Birmingham, UK

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Robotic Process Automation (RPA) [27] has garnered substantial interest as a means of automating repetitive and labor-intensive business processes through software bots. Its adoption spans various industries, including customer service, finance, human resources, supply chain management, and healthcare with the aim of enhancing operational efficiency, minimizing costs and errors, and improving overall customer experience [1, 2, 9, 11, 12, 23, 25, 31]. Despite its popularity, scientific literature on RPA is limited, with existing sources mainly focusing on its features and benefits [10, 13, 15, 17, 21, 27]. Current RPA systems have inherent limitations concerning decision-making, language comprehension, and visual capabilities, as they are designed to adhere to pre-defined rules and workflows using pixel-level encoding of the graphical user interface (GUI)[3, 22, 29]. These functionalities are typically implemented through drag-and-drop interfaces, screenplay recording, or automation frameworks such as selenium[26]. Consequently, these systems lack flexibility in adapting to changes in the UI and struggle to handle tasks that require intricate visual analysis and natural language understanding.

Recent years have witnessed remarkable progress in deep learning and computer vision, leading to advancements in object recognition, image segmentation, and video analysis [6, 14, 16]. Additionally, the introduction of pre-trained large language models such as GPT-3 [4], ChatGPT [5], Llama [28], and PaLM [7] has revolutionized natural language processing, enabling advanced language understanding and generation capabilities. For instance, AI agents such as AgentGPT [1] and AutoGPT [2] can automate a wide range of tasks, including writing, translation, and content generation. Moreover, the advent of Visual Language Models (VLMs) such as Control-Net [34] and Visual-ChatGPT [32], combining text-based LLMs with visual understanding, has opened new avenues for image analysis and processing. While VLMs such as Visual-ChatGPT [32] and Google's Bard [3] can perform tasks such as generating images from textual input, providing image descriptions, and answering questions about images, they require fine-tuning on Web GUIs datasets to identify and localize screen elements in application GUIs. Further, the recently announced GPT-4 [19] by OpenAI has received significant attention due to its promising capabilities in handling multimodal data. However, as of now, GPT-4 has not been released publicly, and its utility and limitations in handling visual data are yet to be evaluated.

These breakthroughs have opened up new possibilities for integrating LLMs with RPA systems towards enabling them to perceive and autonomously interact with complex web applications. For example, Wang et al.[30] conducted a study exploring the use of

[1]AgentGPT: https://github.com/reworkd/AgentGPT
[2]AutoGPT: https://github.com/Significant-Gravitas/Auto-GPT
[3]Google's Bard: https://bard.google.com/

pre-trained language models (LLMs) to enable conversational interaction on mobile user interfaces (UIs). Their research involved providing GUIs to LLMs that were pre-trained for natural language understanding, along with employing various techniques to prompt the LLMs to perform conversational tasks. In another study[18], Pedro et al. utilized the Yolo object detector [24] to identify screen elements such as menus and buttons. However, the study did not propose a method for determining the necessary actions to perform a specific task based on the identified screen elements. Additionally, the training of the object detector was limited to detecting Eclipse IDE screen elements only, requiring the development of a new detector in case of changes in the application type.

To address the limitations of current RPA systems, we propose a novel AI-based RPA system called SmartFlow that uses LLMs coupled with deep-learning based image understanding. It integrates vision capabilities with natural language processing techniques to adapt to changes in the graphical user interface (GUI) and automatically generate navigation workflows. By utilizing vision techniques, SmartFlow identifies and locates screen elements, while the HTML source code provides information about the type of these elements. A large language model such as GPT-3 is then employed to generate navigation workflows based on this information. This navigation workflow is then executed using a scripting language to complete the assigned task. One notable benefit of SmartFlow is its ability to handle diverse application layouts and screen resolutions efficiently. In summary, our paper presents the following contributions:

- We propose an AI-based RPA system called SmartFlow which utilizes LLMs in tandem with deep-vision and is capable of autonomously executing user-assigned tasks.
- SmartFlow leverages HTML code, visual and natural language understanding to interpret the layout mapping. This includes associating field names, their types, and corresponding placeholders/edit fields.
- SmartFlow is designed to be adaptable to GUI changes and handle complex tasks effectively. It achieves this by generating navigation workflows using vision and large language models (LLMs), without relying on predefined pixel-encoded rule-based workflows
- We demonstrate SmartFlow's proficiency in handling multipage form submission applications with diverse field types, such as date pickers, dropdown menus, etc. through the use of vision-based algorithms.
- To demonstrate the effectiveness of SmartFlow, we have curated a dataset called RPA-Dataset, containing generic web applications with various layouts. We intend to release this dataset publicly to foster research in this field.

2 OVERVIEW

Our objective is to automate the generation of navigation workflows for specific tasks within a graphical user interface (GUI) application. Using deep-vision and natural language understanding, we identify screen elements such as field names, placeholders/edit fields, and hints. Subsequently, LLM is used to determine the necessary actions to fill in the required information which are then executed using a scripting engine. Finally, SmartFlow provides updates on the status of the executed task. For example, let's consider the task of registering a new patient in a Hospital system. Traditionally, this process involved manual data entry from a handwritten document. However, we can automate it by digitizing the document using information extraction techniques [8, 20, 33]. SmartFlow then automatically fills in the patient's details in the registration system, eliminating the need for manual data entry.

3 SYSTEM DESIGN: SMARTFLOW

We envisage the following four user-classes for SmartFlow:

- *End-user*: provides all the necessary information, such as task request data, via email or chat-bot to the application, with the objective of having the task executed automatically.
- *Information Validation System (IVS)*: ensures that the task-request received from the end-user is complete and includes all the required information for filling in the data fields necessary to complete the task and adds the task-request to the incoming task directory.
- *Admin*: is responsible for setting up and configuring SmartFlow initially for an application. This involves providing meta-data such as the website URL and HTML source code for all its pages. The Admin also performs layout mapping, which associates visible field names on the application screen with their respective edit-fields and data-hints. We propose two vision-based methods for automatic layout mapping, which are validated by the Admin. In case of any errors, a demonstration approach is used for accurate layout mapping.
- *SmartFlow API*: is responsible for sequentially handling task requests from the incoming requests directory. Upon completion, the API returns the task's output status (e.g., success, failure, or errors) to the task-status directory. The Admin is then responsible for communicating the task statuses to end-users through their preferred communication channel.

Next, we will present three different methods for **Layout Mapping**, each offering unique advantages.

1. **Rule-Based Approach:** After analyzing multiple web application forms, we observed a consistent pattern where field names are usually aligned to the left or top of the edit field, while data-hints are commonly positioned at the bottom or right side. Leveraging these observations, we have devised an automated layout mapping technique that combines vision-based methods with predefined rules and heuristics.
2. **Virtual Grid Approach:** Typically, Layout Mapping Models (LLMs) face challenges in interpreting pixel coordinates to comprehend spatial layouts accurately. Hence, we propose condensing the original layout by converting pixel coordinates into a virtual grid space. Each unit in the virtual grid covers multiple pixel blocks, simplifying misalignment checks to eight neighboring cells and reducing the spatial complexity. We represent the spatial layout using the .CSV format in virtual grids, which are fed as input to the LLM along with a text prompt to generate the layout mapping.
3. **Demonstration by Admin:** If a rule-based or virtual grid-based approach does not yield accurate layout mapping, a demonstration-based approach can be employed. This method involves the administrator providing a demonstration by entering dummy data into the web application form and submitting a JSON file with relevant field information.

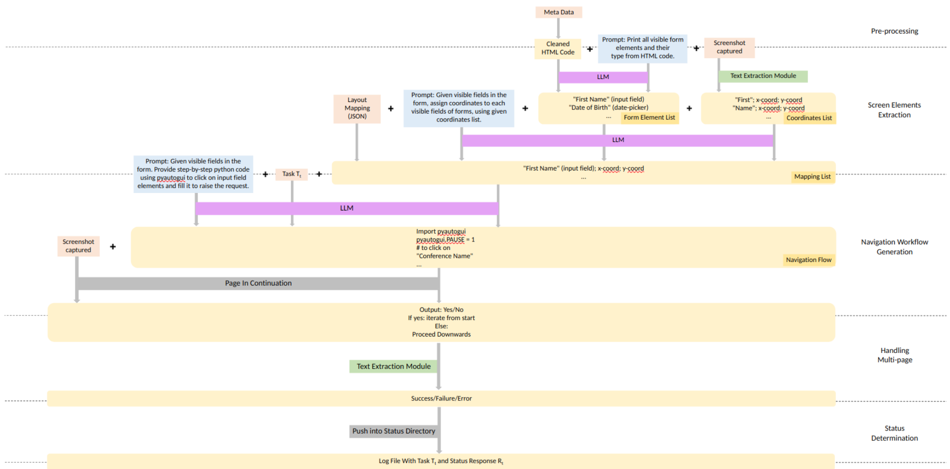

Figure 1: Pipeline of Smartflow

SmartFlow uses visual analysis of the filled and unfilled form images, along with the JSON data, to establish connections between field names, placeholders, and values, ensuring 100% accurate layout mapping.

SmartFlow Algorithm: The Smartflow begins processing as shown in Figure 1, by sequentially extracting and handling requests from the incoming request queue.

- *Pre-processing*: It cleans the HTML source code provided in the input metadata (application URL, HTML source code and layout mapping), ensuring it meets the size limit of Large Language Models. This involves removing unnecessary attributes and classes that could hinder LLM processing.
- *Form Elements Extraction*: The cleaned HTML source code is input into Large Language Models (LLMs) such as GPT-3 [4] and ChatGPT [5] to extract field names and types using the prompt as shown in Figure 1. Meanwhile, a screenshot image of the application is captured, and text-regions are extracted using EasyOCR[4]. The extracted information is merged with the layout mapping to create a Mapping List, which includes field names, types, and coordinates. This Mapping List serves as the textual representation of the visual screen for generating the navigation workflow.
- *Navigation Workflow Generation*: In this step, the Mapping List and task-request are given as input to the LLM with a prompt to generate PyAutoGUI [5] code. This scripting code determines the sequence of actions, including clicking on the correct form-field, to complete the task-request accurately. The precision is crucial to avoid incorrect form submission. The algorithm executes micro-level steps with high accuracy and handles different field types such as date pickers, drop-down menus, upload buttons, radio buttons, and checkboxes using vision-based algorithms invoked by the LLM. More details can be found in the supplementary material.
- *Handling Multi-page Form Submission*: After executing the navigation workflow using the scripting code, SmartFlow captures another screenshot to handle multi-page forms effectively. By leveraging visual cues from the website's layout, it recognizes the continuation of the form and sequentially processes the user's requests to fill in any remaining fields.
- *Determining the status of executed task-requests*: SmartFlow employs a frame difference technique to extract feedback messages related to the success, failure, or errors encountered during form submission. These messages, obtained using a text-extractor, can address network connectivity, missing fields, or successful submissions. By logging these messages into a status queue, SmartFlow facilitates analysis and improves the user experience.

[4]EasyOCR: https://github.com/JaidedAI/EasyOCR
[5]PyAutoGUI Documentation: https://pyautogui.readthedocs.io/en/latest/

4 DATASET DETAILS

To evaluate the effectiveness of integrating vision and large language models (LLMs) in RPA systems, we created the RPA-Dataset. It consists of five web applications, each with five diverse layouts and five user-task requests. The applications cover generic enterprise domains such as Conference Attendance System, New Patient Registration, Sales Lead Generation, Customer Complaint Handling, and Passport Registration. The RPA-Dataset includes the source HTML codes of the applications, along with ground-truth annotations for tasks such as OCR (Optical Character Recognition), Layout Mapping, filling data fields, and handling complex fields such as dropdowns, datepickers and radio-buttons/checkboxes.

5 RESULTS AND DISCUSSIONS

Evaluation Metric: To evaluate SmartFlow's accuracy in generating navigation workflows and entering correct values into data fields, we calculate the following metrics:

- *Text-extraction Accuracy*: Measures the accuracy of detecting text fields on the application screen using OCR techniques such as EasyOCR in terms of Character Error Rate (CER) and Word Error Rate (WER).
- *Layout Mapping Accuracy*: Evaluates the correct association of field names with edit fields, placeholders, and data hints.

Table 1: Table showing the performance of SmartFlow on CAS application.

Layout No.	Page No.	Accuracy						Task completion average time (in mins)	Complex Component Accuracy		
		OCR		Layout Mapping		Filled Data	Request Submission		Datepicker	Dropdown	Radio/Checkbox
		CER	WER	Rule-based	Virtual-Grid						
1	1	0.005	0.087	1.0	0.91	0.91	1.0	4.27	1.0	1.0	0.8
	2	0.004	0.050	1.0	1.0	1.0					
2	1	0.045	0.176	0.8	0.91	0.91	1.0	6.4	1.0	1.0	0.8
	2	0.004	0.025	1.0	1.0	1.0					
3	1	0.037	0.154	0.9	0.91	0.91	1.0	6.7	0.933	1.0	0.8
	2	0.008	0.075	1.0	1.0	1.0					
4	1	0.016	0.091	1.0	0.91	0.91	1.0	6.8	1.0	1.0	0.8
	2	0.005	0.054	1.0	1.0	1.0					
5	1	0.032	0.107	1.0	0.91	0.98	1.0	4.8	1.0	1.0	0.0
	2	0.005	0.027	1.0	1.0	1.0					
Average		**0.015**	**0.086**	**0.97**	**0.955**	**0.95**	**1.0**	**5.7**	**0.98**	**1.0**	**0.64**

Table 2: Table showing the performance of SmartFlow across different applications with diverse layouts. We report average accuracy of all the layouts per application.

Application	Average Accuracy						Task completion average time (in mins)	Complex Component Average Accuracy		
	OCR		Layout Mapping		Filled Data	Request Submission		Datepicker	Dropdown	Radio/Checkbox
	CER	WER	Rule-based	Virtual-Grid						
CAS	0.015	0.086	0.97	0.955	0.95	1.0	5.7	0.98	1.0	0.64
Patient Registration	0.0	0.0	0.92	0.876	0.952	1.0	1.952	0.96	1.0	0.88
Sales Lead Generation	0.015	0.039	0.92	0.841	0.887	1.0	1.55	-	1.0	0.50
Customer Complaint	0.008	0.029	0.964	1.0	0.913	1.0	1.36	1.0	1.0	0.873
Passport Application	0.009	0.038	0.928	0.986	0.963	1.0	1.604	0.96	1.0	0.86
Average	**0.009**	**0.038**	**0.94**	**0.931**	**0.933**	**1.0**	**1.433**	**0.985**	**1.0**	**0.75**

- *Filled Data Accuracy*: Determines the accuracy of filling fields in the application form with correct data values.
- *Request Submission Accuracy*: This metric measures the success or failure of executing the task request.
- *Complex Component Accuracy*: Reports the accuracy of filling data in complex fields such as datepickers, dropdowns, radio buttons and checkboxes.
- *Task Completion Time*: This measures the time (in minutes) taken to complete one specific task-request.

Experimental Results: We conducted our experiments using OpenAI's LLM GPT-3 [4] API, which is publicly available. The experiments were performed on a GTX 1080 machine with 8 GB GPU Memory. In Table 1, we present the performance results of SmartFlow on the Conference Attendance System (CAS), a two-page web application with diverse layouts. The text extraction accuracy of OCR is high, with an average CER of 0.015 and WER of 0.086. We also compare the accuracy of the rule-based and virtual-grid layout mapping approaches, which show similar and satisfactory results. The minor mistakes in layout mapping can be attributed to certain factors such as closeness of field name and/or hint with incorrect edit-field, cascaded OCR text detection error. These errors were corrected during the initial setup of SmartFlow on the system by Admin. The accuracy of filled data is 95%, with errors primarily occurring in radio-button and checkbox fields. Finetuning LLMs to handle these fields would significantly improve the accuracy of filled data. The request submission accuracy is 100%, indicating that SmartFlow accurately reads the status of executed requests. The average task completion time for CAS is 5.7 minutes, considering its multi-page nature. Variations in task completion time across different layouts and user-tasks are mainly influenced by datepicker selections and scrolling within dropdown fields.

Next, we report the average accuracy for each application of the RPA-dataset with different layouts in Table 2. For more detailed results, please refer to the supplementary file [6]. In Table 2, it is evident that SmartFlow efficiently automated various applications with an average filled data accuracy of 93.3% and an average time to submit requests of 1.433 minutes. The main challenge lies in accurately selecting options for radio-buttons and checkboxes which can be achieved by fine-tuning the LLMs with such data fields.

[6]SmartFlow: https://github.com/arushijain45/SmartFlow

6 LIMITATIONS OF SMARTFLOW

In this section, we mention the limitations of SmartFlow:

- Dynamic fields: In the current version, handling dynamic fields is not supported. One approach is to generate PyAutoGui code for each field individually and perform layout mapping after filling data in that field. However, this method is time-consuming and inefficient. We are actively researching more efficient solutions for dynamic field handling.
- *Scrollable forms*: We can use the HTML source code to determine if the page is scrollable and then capture screenshots of the visible fields, perform layout mapping and generate PyAutoGui code for filling them. If there are hidden fields or buttons such as "next" or "submit," we scroll down the page and repeat the process until we locate the submit button.
- *Inference of field types*: Future versions of SmartFlow aim to enhance the accuracy and flexibility of determining field types by training a deep learning-based object detector. This will reduce reliance on the HTML source code alone. However, due to the unavailability of training data, this feature is not included in the current version.

7 CONCLUSION AND FUTURE WORK

This paper introduced SmartFlow, an AI-driven RPA system designed to autonomously execute user task-requests. By integrating computer vision and generative models such as LLMs, SmartFlow is able to automatically generate navigation workflows and adapt to variations in GUIs and applications without human intervention. Our experiments on a self-created RPA-dataset, consisting of diverse web applications with varying layouts and user task-requests, showcased the impressive performance of SmartFlow. Moving forward, our future work will focus on handling dynamic web-applications with scrollable forms and training a deep-learning-based object detector to infer field types, reducing the reliance on HTML code.

REFERENCES

[1] Santiago Aguirre and Alejandro Rodríguez. 2017. Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study. In *Workshop on Engineering Applications*.

[2] Sorin Anagnoste. 2018. Robotic Automation Process – The operating system for the digital enterprise. *Proceedings of the International Conference on Business Excellence* 12 (2018), 54 – 69.

[3] Automation Anywhere. 2003. The automation success platform. https://www.automationanywhere.com/. Accessed: 2023-05-15.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Muthukumar Subbiah, Jared Kaplan, Prafulla Dhariwal, Kalyan Neelakantan, Dario Amodei, and Ilya Sutskever. 2022. ChatGPT: Optimizing Language Models for Dialogue. *arXiv preprint arXiv:2201.08237* (2022).

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. *CoRR* abs/2005.12872 (2020). arXiv:2005.12872 https://arxiv.org/abs/2005.12872

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs.CL]

[8] Vishwanath D, Rohit Rahul, Gunjan Sehgal, Swati, Arindam Chowdhury, Monika Sharma, Lovekesh Vig, Gautam Shroff, and Ashwin Srinivasan. 2018. Deep Reader: Information extraction from Document images via relation extraction and Natural Language. arXiv:1812.04377 [cs.CV]

[9] Christian Flechsig, Franziska Anslinger, and Rainer Lasch. 2022. Robotic Process Automation in purchasing and supply management: A multiple case study on potentials, barriers, and implementation. *Journal of Purchasing and Supply Management* 28, 1 (2022), 100718. https://doi.org/10.1016/j.pursup.2021.100718

[10] Jerome Geyer-Klingeberg, Janina Nakladal, Fabian Baldauf, and Fabian Veit. 2018. Process Mining and Robotic Process Automation: A Perfect Match. In *International Conference on Business Process Management*.

[11] Pooja Gupta, Semila Fernandes, and Manish Jain. 2018. Automation in recruitment: a new frontier. *Journal of Information Technology Teaching Cases* 8 (2018), 118–125.

[12] Petri Hallikainen, Riitta Bekkhus, and Shan Pan. 2018. How OpusCapita Used Internal RPA Capabilities to Offer Services to Clients. *MIS Q. Executive* 17 (2018).

[13] Lucija Ivanić (Juro), Dalia Sua Vugec, and Vesna Bosilj-Vuksic. 2019. Robotic Process Automation: Systematic Literature Review. In *International Conference on Business Process Management*.

[14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything. arXiv:2304.02643 [cs.CV]

[15] Abderrahmane Leshob, Audrey Bourgouin, and Laurent Renard. 2018. Towards a Process Analysis Approach to Adopt Robotic Process Automation. *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)* (2018), 46–53.

[16] Junwei Liang. 2021. From Recognition to Prediction: Analysis of Human Action and Trajectory Prediction in Video. arXiv:2011.10670 [cs.CV]

[17] Somayya Madakam, Rajesh M. Holmukhe, and Durgesh Kumar Jaiswal. 2019. The Future Digital Work Force: Robotic Process Automation (RPA). *Journal of Information Systems and Technology Management* (2019).

[18] Pedro Martins, Filipe Sá, Francisco Morgado, and Carlos Cunha. 2020. Using machine learning for cognitive Robotic Process Automation (RPA). In *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*. 1–6. https://doi.org/10.23919/CISTI49556.2020.9140440

[19] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[20] Rasmus Berg Palm, Florian Laws, and Ole Winther. 2021. Attend, Copy, Parse – End-to-end information extraction from documents. arXiv:1812.07248 [cs.CL]

[21] Esko Penttinen, Henje Kasslin, and Aleksandre Asatiani. 2018. How to Choose between Robotic Process Automation and Back-end System Automation?. In *European Conference on Information Systems*.

[22] SS&C Blue Prism. 2001. Intelligent Robotic Process Automation. https://www.blueprism.com/. Accessed: 2023-05-15.

[23] M. Ratia, Jussi Myllärniemi, and Nina Helander. 2018. Robotic Process Automation - Creating Value by Digitalizing Work in the Private Healthcare? *Proceedings of the 22nd International Academic Mindtrek Conference* (2018).

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640 [cs.CV]

[25] Mário Romao, Joao Costa, and Carlos J. Costa. 2019. Robotic Process Automation: A Case Study in the Banking Industry. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. 1–6. https://doi.org/10.23919/CISTI.2019.8760733

[26] Monika Sharma. 2014. Selenium Tool: A Web based Automation Testing Framework.

[27] Julia Siderska. 2020. Robotic Process Automation — a driver of digital transformation? *Engineering Management in Production and Services* 12 (2020), 21 – 31.

[28] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023).

[29] UiPath. 2005. AI-powered business automation. https://www.uipath.com/. Accessed: 2023-05-15.

[30] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling Conversational Interaction with Mobile UI using Large Language Models. arXiv:2209.08655 [cs.HC]

[31] Leslie P. Willcocks, Mary C Lacity, and Andrew Craig. 2017. Robotic process automation: strategic transformation lever for global business services? *Journal of Information Technology Teaching Cases* 7 (2017), 17–28.

[32] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. arXiv:2303.04671 [cs.CV]

[33] Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. 2020. PICK: Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks. arXiv:2004.07464 [cs.CV]

[34] Lvmin Zhang and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. arXiv:2302.05543 [cs.CV]