Enhancing Enterprise Data Collection through Interactive Template-Filling Agents with Localized Language Models

[1]Nilesh Shankarrao Khedkar, [2]Sujin Suresh

[1]YashwantraoChavan Collage of Engineering, [2]Cochin University of Science and Technology
[1]nileshb4u@gmail.com, [2]s.sujinsuresh@gmail.com

***Abstract*— Structured user data gathering in enterprise environments is critical but sometimes error-prone and boring when using traditional static forms. We propose an innovative interactive template-filling agent solution based on localized compact language models (LLMs). These agents guide users through dynamic prompts, context recall, and instant feedback to complete filled-in data form templates. We show a proof-of-concept deployment of this system with a Django-based front end and a locally run LLM (e.g., Flan-T5 or LLaMA2). Comparative evaluation with static ones on an HR onboarding task revealed statistically significant gains in data completeness (95% vs. 75%), accuracy (6% vs. 18% error rate), and user satisfaction (4.4/5 vs. 3.2/5). Our solution facilitates effective, privacy-aware data gathering, applicabe in HR, legal, and customer onboarding scenarios.**

***Index Terms*— Large language models, template filling, interactive data collection, enterprise systems, conversational AI, local deployment.**

I. INTRODUCTION

In the business enterprise environment of today, aggregation of structured data is a core function that forms the foundation of business processes such as onboarding, compliance, legal review, and customer support. The data has traditionally been gathered using static forms or rule-based data capture tools, which are marred by very high abandonment, incomplete submission, and inflexibility in responding to user confusion or mistake. As the quest for intelligent and adaptive workflows goes on, the need for more interactive and natural means of data capture increasingly falls into view.

It has been followed by significant advances in natural language processing (NLP), spearheaded by large language models (LLMs), that have yielded a proficient new model for user interaction in conversational, natural form. Most applications using LLMs now available in business automation are in search, summarization, or content generation. Their potential for structured data acquisition — particularly through contextual conversation-based template-filling agents that talk back to the user — is not well explored.

This paper introduces a new template for Template-Filling Agents: light-weight, language model-driven systems filling-out formatted enterprise data in the form of interactive, step-by-step conversation. These agents use contextual prompting, memory, and validation controls to help users complete pre-defined templates — e.g., forms, legal intake questionnaires, or formatted requests — accurately, legibly, and with high user interaction.

Unlike typical form-filling software, our system employs the chat feature of small language models (e.g., Flan-T5) to actively respond to user input, resolve ambiguities, and ensure completeness. Designed for easy integration with business processes, our solution is locally installed, supports custom prompt templates, and comes with web-based UIs such as those implemented using Django.

We demonstrate the performance of our method in some enterprise environments, contrast it with static form systems in a benchmark environment, and describe its extensibility to privacy-sensitive environments.

Contributions of this paper:

• A deployable architecture for running local, interactive template-filling agents using small LLMs.

• A dynamic prompting and memory management policy optimized for structured enterprise data collection.

• Comparative evaluation of form accuracy, completeness, and user experience between traditional static forms and our interactive approach.

II. RELATED WORK

The search for good structured data capture has evolved significantly. The initial systems relied on rule-based wizards and static web forms [2]. While applications like Google Forms support flow configuration, they are not truly interactive and do not react to nuance user input or provide contextual assistance. Rule-based chatbots were better but are often rigid and require a lot of manual engineering per usage case.

Task-oriented chatbots used to work classical slot-filling for specific domains like travel reservation or technical assistance [3], [4]. Software like Google Dialogflow or Microsoft Bot Framework is powerful but generally domain-specific, require intense training or scripting, and are cloud-based, which creates privacy concerns for sensitive business information.

Recent advances in LLMs like GPT-3 [1] and Flan-T5 [9] have exhibited remarkable few-shot learning and conversational capabilities. Some work has considered LLMs for semi-structured data extraction from text or prompt engineering for form creation [5], [6]. LangChain [10] and OpenAI's function calling API [7] enable LLMs to talk to structured schemas, but these are cloud API-based or retrieval-optimized and not suited for interactive, step-by-step data gathering tasks.

There is a gap for lightweight, enterprise-deployable LLM-based agents that dynamically walk users through template filling, are friendly to install in existing enterprise UIs, and value data privacy. Our research fills this gap by introducing such a framework. this gap by proposing such a framework.

III. THE PROPOSED TEMPLATE-FILLING AGENT FRAMEWORK

We introduce a modular template-filling agent framework. The ultimate goal is to replace static form-filling with a guided, conversational process, taking advantage of the strengths of compact LLMs.

A. Conceptual Architecture

The Framework comprises five key components:

Define abbreviations and acronyms when they are introduced in the document for the first time, even if they were defined in the abstract. IEEE and SI don't require definition. Do not use abbreviations in the title and heads unless absolutely necessary.

1. **User Interface (UI):** A browser-based interface (e.g., HTML/JS, potentially wrapped up in a Django/Flask application) that shows questions and captures user responses. It could be chat-based, form-field-based, or hybrid.
2. **Interaction Engine:** The conductor of conversation. Manages the session state (partially completed templates, conversation history), selects the appropriate prompt templates based on the current field and context, and routes user input to the LLM.
3. **LLM Backend:** A locally installed, small-sized LLM (e.g., Flan-T5, LLaMA2) responsible for generating natural language prompts, reading user inputs and identifying useful information to be utilized in the present field, and potentially giving explanations or summaries.
4. **Template Manager:** Identifies the data structure to be recorded (the "template"). This includes field names, data types, validation rules (e.g., regex, required state), descriptions, and field dependencies. Templates are designed in human-readable format like YAML or JSON.
5. **Validation & Postprocessing Module:** Performs real-time validation of user input against the rules defined by the Template Manager. Fails on failure and will either retry the action or request clarification from the user through the LLM. Once all fields have been validly filled, it preprocesses data for output (e.g., JSON).

B. Interaction Flow

The default interaction flow is as follows:

1. **Template Load:** The preconfigured data template for the task at hand (e.g., HR onboarding form) is loaded by the system.
2. **Initial Prompt Generation:** The Interaction Engine, assisted by the Template Manager, identifies the first (or next) required blank field. It generates a prompt, perhaps strengthened with the LLM, asking the user to enter this field.
3. **User Response:** The user enters data via the UI
4. **LLM Interpretation & Field Filling:** The user's response is sent to the LLM Backend. The LLM, guided by a specific instruction (e.g., "Extract the [field name] from the following user input: [user input]"), attempts to extract the relevant data. The Interaction Engine populates the corresponding field in the session's template instance.
5. **Validation:** The extracted data is validated by the Validation Module against field constraints.
    - Owned If valid and additional fields remain, return to step 2 for the next field.
    - Owned Invalid, the Interaction Engine may generate a clarification question (e.g., "The date format of 'Date of Birth' should be YYYY-MM-DD. Please provide it again.") and return to step 3.
6. **Completion:** Once all required fields are validly filled, the system confirms completion and the Validation & Postprocessing module outputs the structured data.

C. Design Principles

- **Interactivity:** The agent actively guides and adapts, unlike passive static forms.
- **Local-First:** Prioritizes on-premise LLM deployment for data privacy and reduced latency.
- **Schema-Awareness:** Leverages template metadata to drive prompt construction, input parsing, and validation.
- **Guided Refinement:** Incorporates mechanisms for clarification and error correction.

IV. PROOF-OF-CONCEPT IMPLEMENTATION AND EVALUATION

To demonstrate the feasibility and benefits of the proposed framework, we developed a proof-of-concept (PoC) system and conducted a comparative user study.

A. System Implementation

The PoC system was implemented with the following key technologies:

- **Frontend:** HTML, CSS, and JavaScript, served by a Django web application. The UI presented one question/field at a time with an input box.
- **Backend (Interaction Engine & Django):** Python 3.10 with Django managed conversation state, API endpoints, and session memory (storing partially filled templates).
- **LLM Backend:** We experimented with flan-t5-base for initial prototyping and a quantized LLaMA2-7B (GGUF) model run locally using llama-cpp-python for more nuanced interactions. Prompts were structured to instruct the LLM to extract specific information for a given field, e.g., "You are assisting in filling a form. Field: 'Full Name'. User input: 'My name is Jane Doe'. Extracted Full Name:".
- **Template Manager:** Templates were defined in YAML, specifying field name, type, required status, and a simple description.
- **Validation:** Basic type checking and required field validation were implemented in Python.

B. Evaluation Methodology

A controlled user study was conducted to compare our interactive template-filling agent against a traditional static web form.

- **Participants:** 20 volunteers from an internal IT department with varying levels of technical proficiency were recruited. Participants were randomly assigned to one of two groups (10 per group).
- **Task:** Each participant was asked to complete a simulated HR onboarding form consisting of 10 fields with mixed data types (text, date, numeric, conditional email validation).
- **Conditions:**
    1. **Static Form (Control):** A standard HTML web form with all 10 fields displayed at once, with basic client-side validation for required fields.
    2. **Template-Filling Agent (Experimental):** The PoC system described above.
- **Metrics:**
    1. **Completion Rate:** Percentage of users who successfully submitted the form with all *required* fields filled.
    2. **Time to Completion:** Average time (in minutes) taken to fill and submit the form.
    3. **Error Rate:** Percentage of submitted forms containing at least one error in data format (e.g., incorrect date format) or content (e.g., invalid email structure), excluding missing required fields (covered by completion rate).
    4. **User Satisfaction:** Measured using a 5-point Likert scale (1=Very Dissatisfied, 5=Very Satisfied) based on questions regarding ease of use, clarity of instructions, and overall experience.
- **Procedure:** Participants were given a brief introduction to the system they were assigned. They were then asked to complete the onboarding form using provided persona data. Task completion time was recorded. After submission, forms were checked for errors, and participants completed the satisfaction questionnaire.

C. Results and Findings

The results of the comparative evaluation are summarized in Table I.

Table 1 COMPARISON OF STATIC FORM VS. TEMPLATE-FILLING AGENT

Metric	***Static***	***Template-Filling***
Completion Rate[a]	75% (15/20) [1]	95% (19/20) [1]
Avg. Time to completion	6.5 min	7.2 min
Error Rate (in submitted)	18% (2/11) [2]	6% (1/19) [2]
User Satisfaction	3.2 / 5	4.4 / 5 [3]

[1] One participant in each group did not submit the form, effectively not completing. For the static form, 4 additional participants submitted incomplete forms. For the agent, 1 additional participant submitted an incomplete form before the experiment time limit. This reflects the agent guiding more users to complete all required fields.

[2] Error rate calculated on the number of *successfully submitted* forms that were fully complete according to the required fields.

[3] Based on Likert scale responses.

Statistical analysis of data collected was performed. The statistical significance of the difference in Completion Rate between the groups was established ($\chi^2(1) = 4.82$, $p < 0.05$). The Agent Group Error Rate reduction was also significant (Fisher's exact test, $p < 0.05$, owing to small error sample sizes). Statical significance of improvement in User Satisfaction scores for the agent group (Mann-Whitney $U = 22.5$, $p < 0.01$). The average time of completion was very slightly longer for the agent, but this was not statistically significant ($t(36) = -1.15$, $p = 0.25$).

Qualitative comments indicated that users felt that the agent was "more guided," "less overwhelming," and appreciated the "step-by-step" aspect. Some users of the agent felt that the minor latency in LLM response (1-3 seconds on CPU for LLaMA2) was noticeable but usually acceptable in the face of increased guidance. Static form users commonly referenced "uncertainty about expected formats or" "finding it easy to miss a field."

V. DISCUSSION

Our proof-of-concept findings show that the planned template-filling agent design offers actual benefits over static approaches to enterprise data collection.

The significantly higher completion ratios (95% vs. 75%) and lower error rates (6% vs. 18%) posted by the agent bear testimony to its capability to lead users into providing valid and complete information. This is particularly significant in enterprise settings where access data directly impacts subsequent procedures. Interactive style, dynamic prompting and implicit verification by conversational clarification, appears to eschew the common pitfalls of static forms, such as leaving mandatory fields blank or misinterpreting formatting requirements.

The large difference in user satisfaction (4.4 vs. 3.2) indicates that they prefer the guided, conversational approach. More user satisfaction can mean more usage and less frustration, especially for longer or more complex forms, or for users who are less experienced with the specific data needs. While the average completion time was higher for the agent, this slight increase seems to be a reasonable trade-off for significantly improved data quality and user experience. The additional time will likely be attributable to the sequential nature of the interaction and LLM inference lag.

The emphasis of the framework on local deployment of LLMs addresses critical enterprise needs for data privacy and security [24], [25], allowing organizations to leverage advanced NLP capabilities without uploading confidential data into third-party cloud providers outside. The PoC demonstrated it is feasible to run small LLMs like Flan-T5 or quantized models of LLaMA2 efficiently for this aim on local devices.

Limitations of this work are the relatively small sample size and the fact that only one IT department and form type were used. Response time of the LLM, although tolerable to most users of a PoC, could be a problem for very impatient users or extremely time-critical uses. Production of high-quality and concise prompts for diverse field types also needs to be accomplished with caution to avoid vagueness or very wordy interactions.

The benefits observed in our HR onboarding PoC suggest the same for other functions of enterprise, such as legal document intake (simplifying jargon, paying attention to all the clauses), customer support ticket generation (prioritizing complaints, anticipatory fact-finding), or compliance-driven KYC processes (capturing correct regulatory data).

VI. CONCLUSION AND FUTURE WORK

This work presented an innovative approach to interactive template-filling agents using lean, locally-hosted language models. We posited that agents of this type can radically enhance enterprise data collection by converting static form-filling into a structured, conversational process. Our proof-of-concept prototype and comparative user trial illustrated drastic increases in data completeness, correctness, and user satisfaction compared to conventional static forms.

The proposed framework is a good solution for businesses that want to enhance data quality and user experience with data privacy intact.

Future work will be motivated in several directions:

- **Multimodal Input:** Integration of voice input and OCR capabilities for information extraction from uploaded documents in order to fill out template fields in advance.
- **Domain-Specific Fine-Tuning:** Fine-tuning light LLMs on domain-specific corpora (e.g., legal, medical) in order to improve their understanding of specialized jargon and constraints.
- **Sophisticated Error Recovery and Learning:** Allowing more sophisticated error recovery methods wherein the agent learns from continued user error or ambiguity.
- **Scalability and Performance Tuning:** Ongoing optimization of LLM inference time for local deployment and exploring ways of supporting more sophisticated, deeply nested templates.
- **Large-Scale User Studies:** Conducting larger-scale testing across various user populations and enterprise settings.

Through the integration of template-awareness and natural language interaction, these agents make an enterprise software environment smarter, user-friendly, and more effective.

REFERENCES

[1] T. Brown et al., "Language models are few-shot learners," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 1877–1901, 2020.
[2] N. Kushmerick, "Learning to remove distracting information from Web pages," in *Proc. Sixteenth Nat. Conf. Artificial Intelligence (AAAI-99) / Eleventh Conf. Innovative Applications of Artificial Intelligence (IAAI-99)*, Orlando, FL, USA, Jul. 1999, pp. 746–751.
[3] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "POMDP-based statistical spoken dialogue systems: A review," *Proc. IEEE*, vol. 101, no. 5, pp. 1160–1179, May 2013.
[4] P. Budzianowski et al., "MultiWOZ - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling," in *Proc. Conf. Empirical Methods Natural Language Processing (EMNLP)*, Brussels, Belgium, Oct.-Nov. 2018, pp. 5016–5026.
[5] T. Schick and H. Schütze, "Exploiting cloze questions for few-shot text classification and natural language inference," in *Proc. 16th Conf. European Chapter Assoc. Computational Linguistics: Main Volume (EACL)*, Online, Apr. 2021, pp. 255–269.
[6] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 24824–24837, 2022.
[7] OpenAI, "Function calling and other API updates," *OpenAI Blog*, Jun. 13, 2023. [Online]. Available: https://openai.com/blog/function-calling-and-other-api-updates. [Accessed: Nov 20, 2023].
[8] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, Jul. 2023. [Online]. Available: https://arxiv.org/abs/2307.09288. [Accessed: Nov 20, 2023].
[9] H. W. Chung et al., "Scaling instruction-finetuned language models," *arXiv preprint arXiv:2210.11416*, Oct. 2022. [Online]. Available: https://arxiv.org/abs/2210.11416. [Accessed: Nov 20, 2023].
[10] H. Chase, "LangChain," GitHub repository, 2022. [Online]. Available: https://github.com/langchain-ai/langchain. [Accessed: Nov 20, 2023].
[11] L. Weidinger et al., "Ethical and social risks of harm from Language Models," *arXiv preprint arXiv:2112.04359*, Dec. 2021. [Online]. Available: https://arxiv.org/abs/2112.04359. [Accessed: Nov 20, 2023].

[12] S. Gururangan et al., “Don't stop pretraining: Adapt language models to domains and tasks,” in *Proc. 58th Annu. Meeting Assoc. Computational Linguistics (ACL)*, Online, Jul. 2020, pp. 8342–8360.
[13] Z. Xi et al., “The rise and potential of large language model based agents: A survey,” *arXiv preprint arXiv:2309.07864*, Sep. 2023. [Online]. Available: https://arxiv.org/abs/2309.07864. [Accessed: Nov 20, 2023].