Developing a Mobile Virtual Assistant using Large Language Models for Task Automation

Dinh Minh Hieu Phan and Ngoc Hoang Luong

University of Information Technology, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam
hieupdm.17@grad.uit.edu.vn, hoangln@uit.edu.vn

1 Prompt Templates

In this section, we provide the detailed descriptions of the prompt templates used in different tasks. Each template is designed to guide the LLM in generating specific outputs based on the given inputs.

Identify App. The following prompt template is used to determine which app the user is referring to based on a list of apps and the user's request:

~~~
Given the following input:
1. **App List**: A list of apps that the user can refer to.
    Example:  "Tiktok, Shoppe, Youtube, XanhSM, Facebook"
2. **Request**: The user request that needs to be completed.
    Example:
    ```
    "I want to watch a video."
    ```
Based on the inputs, determine which app the user is referring to and return the result in JSON format with the following
↪  structure. Don't add any additional information to the response.
Example:
```json
{
    "apps": ["Youtube", "Tiktok"]
}
~~~

Query Similar Flows. This prompt template is used to find similar flows based on a user request and a list of predefined flows. The model identifies the indices of the matching flows and returns them in CSV format:

~~~
Given the following inputs:
1. **Request**: The user request that needs to be completed.
    Example:
    ```
    "Complete the registration process."
    ```
2. **Flows**: A CSV list with the following properties:
    - `flow_desc`: The description of the flow
    - `user_request`: The user request that the flow fulfills
    Each row represents a flow.
    Example:
    ```
    flow_desc,user_request
    Registration process,Complete the registration process
    ```
Based on the input request, determine flows index that could match the user request (maximum 10 indices) and return the result in
↪  CSV format with the following structure.
If there are no similar flows, return -1.
Don't add any additional information to the response.
Example response that has similar flows:
    0,2
Example response that has no flows:
    -1
~~~

Generate UI Element Description. This prompt template is used to generate a description for a UI element based on its properties. The description is returned as a string:

~~~~
    You will be provided with a UI element image and CSV properties including android_class, text, and content_desc. Your task is
   ↪  to generate a description for the content_desc.
    ### Return only a string description
    ### Example Usage:
    **UI element**
    android_class,text,content_desc
    android.widget.Button,Like,
    **Response:**
    "Like button"
~~~~

Generate Screen Description This prompt template is used to generate a description for a screen based on the UI elements it contains. The description is returned as a string:

~~~~
    You will be provided with a UI element image and CSV properties including android_class, text, and content_desc. Your task is
   ↪  to generate a description for the content_desc.
    ### Return only a string description
    ### Example Usage:
    **UI element**
    android_class,text,content_desc
    android.widget.Button,Like,
    **Response:**
    "Like button"
~~~~

Determine Next Action. This prompt template helps in determining the next action to be taken based on user requests, current UI elements, previous actions, and similar flow guides, which contain user_guide to assist in decision-making. The result is returned in JSON format:

~~~~
    Given the following inputs:
1. **Request**: The user request that needs to be completed.
    Example:
    ```
    "Complete the registration process."
    ```
2. **Current UI Elements**: A CSV list with the following properties:
    - `index`: The index of the UI element
    - `android_class`: The Android class of the UI element (e.g., `Button`, `TextView`)
    - `resource_id`: The unique identifier for the UI element
    - `text`: The text displayed on the UI element
    - `content_desc`: The content description of the UI element
    - `x1`, `y1`: The coordinates of the top-left corner of the UI element
    - `x2`, `y2`: The coordinates of the bottom-right corner of the UI element
    Example:
    ```
    index,android_class,text,content_desc,x1,y1,x2,y2
    0,Button,Submit,submit_button,100,200,300,400
    1,TextView,Welcome,description_welcome,50,100,200,150
    ```
3. **Previous Actions**: A CSV list of previously executed actions:
    - `screen_desc`: The screen description of the screen acted upon
    - `content_desc`: The content description of the UI element acted upon
    - `action`: The action performed (e.g., `click`, `input`)
    - `data`: Any data associated with the action
    Example:
    ```
    screen_desc,content_desc,action,data
    register screen,name_field,input,John Doe
    register screen,email_field,input,john.doe@example.com
    ```
4. **Similar Flow Guide**: A list of similar flows with the following properties:
    - `flow_description`: The description of the flow
    - `steps`: The steps to complete the flow in CSV format with the following properties:
        - `screen_desc`: The description of the screen
        - `user_guide`: The user guide to complete the step
        - `element_content_desc`: The description of the element content
        - `action`: The action to perform
    "Note": "Must execute following similar flow guide corresponding to flow_desc",
    Example:
    ```
    [

            "flow_description": "Complete the registration process.",
            "steps": [
            "Registration screen, Enter name,name_field,input",
            "Confirmation screen, Click submit button,submit_button,tap",
            ]

    ]
Based on the inputs, determine the next action to take and return the result in JSON format with the following structure.
Don't add any additional information to the response.:
~~~~

~~~
```json

    "index": "<index of the UI element to act on, if need to action on screen then index will be -1>",
    "action": "<action to perform (tap, input, swipe_up, swipe_down, swipe_left, swipe_right, return)>",
    "additional_data": "<additional data required for the action>",
    "completed": "boolean to indicate if the flow is completed or not after this action"
~~~