https://doi.org/10.15407/csc.2024.03.053
UCD 514.18

V.YU. LEVCHUK, Master's degree, National University of Kyiv-Mohyla Academy,
H. Skovorody str., 2, Kyiv, Ukraine, 04070,
ORCID: https://orcid.org/0009-0001-6613-7478,
pifagor6541@gmail.com

THE UNIVERSAL MODULE FOR INTEGRATION OF AN INTELLIGENT ASSISTANT INTO IOS APPLICATIONS

Investigated current implementations of the integration of intelligent assistants into mobile applications. Identified key disadvantages of existing implementations and formed the criteria for a universal intelligent assistant. Developed a proprietary software module for integrating an intelligent assistant into iOS application, which provides autonomy, minimal resource requirements, and simplifies the development process. Created a photo editor application to test the operation of the software module. The test results were presented and further development prospects were described.

Keywords: *intelligent assistant, artificial intelligence, semantic search, natural language, model, machine learning, speech recognition, graphical interface.*

Introduction

Currently, the field of artificial intelligence (AI) is experiencing a new wave of interest, driven by the rapid development of large language models ("LLMs") and products based on them. Chatbots such as ChatGPT, developed by OpenAI and released to the public on November 30, 2022, can process and generate natural language at an unprecedented level of quality for many languages and fields of knowledge [1]. Because of this, chatbot-based intelligent assistants have become useful tools for performing search and content creation tasks.

However, despite their effectiveness in these areas, such chatbots have a significant limitation: they are unable to interact with the device's operating system and its applications. This is an important drawback, because according to a study of user behavior in devices with virtual assistants [2], a wide range of functionality is one of the most important qualities of an intelligent assistant for users. That is why operating system-level intelligent assistants are very popular [3]. These primarily include Apple Siri for the iOS operating system and Google Assistant for Android. Such assistants provide the ability to integrate with third-party applications but have several significant limitations and disadvantages, which will be disclosed in this article.

The purpose of this article is to demonstrate the current possibilities of creating an intelligent assistant for the iOS mobile operating system with minimal requirements for development resources

Cite: Levchuk V.Yu. The Universal Module for Integration of an Intelligent Assistant into iOS Applications. *Control Systems and Computers*, 2024, 3, 53-59. https://doi.org/10.15407/csc.2024.03.053

and device computing power. This research aims to overcome the existing limitations and create a more efficient and versatile solution.

This study is divided into the following parts: identifying the flaws of existing implementations and defining the criteria for the desired solution; writing a software module that meets the stated criteria; writing a test application with the integrated module; and demonstrating the results of the integration.

Problem Setting

OS-level virtual assistants are most often used on mobile devices. These primarily include Apple Siri for iOS and Google Assistant for Android, which use the same approach to integrate with third-party applications – intents.

This approach involves application developers defining interaction points through which the assistant and the application communicate. These interaction points should belong to the activity categories predefined by the assistant or application developers to typify them and allow the assistant to better identify them. In this way, applications provide the assistant with separate, independent pieces of functionality that have a clearly defined purpose and conditions for execution.

In addition, for most operations, existing virtual assistants use server processing power and therefore depend on an Internet connection.

As a result, existing implementations of intelligent assistants have several significant drawbacks:

1) limited functionality: the assistant interacts with individual application functions, not the entire application;

2) insufficient flexibility: assistants interact best with apps from a limited set of activity categories, such as calendars, messengers, shopping apps, etc.; while for other types of apps, the quality of performance can be significantly worse;

3) complexity of integration: manual configuration of each interaction point requires significant development resources;

4) dependence on the network: the need for an Internet connection makes it impossible to use assistants in many situations and raises privacy concerns among users [4].

Existing intelligent assistants Apple Siri and Google Assistant were created in the conditions of less developed artificial intelligence and weaker hardware, which led to the above mentioned shortcomings. These problems hinder the development of these systems and slow down the introduction of artificial intelligence into everyday tasks.

For this reason, the task of creating a new software solution that provides the functions of an intelligent assistant for mobile phones, taking into account modern advances in hardware and the development of artificial intelligence, is becoming increasingly important. This solution has the form of a software module, with an emphasis on autonomy and ease of integration for the developer, which is demonstrated on the example of integration into a photo editor.

Purpose

This paper aims to develop a universal intelligent assistant integration module that can be easily added to any iOS application. This intelligent assistant allows the user to interact with the application using voice requests, namely to search for and perform the desired functionality based on an approximate description. The implemented module must meet the following requirements:

1) *Universality*: this intelligent assistant can be integrated into any application, regardless of its subject matter, and does not require specific training.

2) *Ease of integration*: the assistant's software interface is as simple and intuitive as possible and does not require much time to master.

3) *Autonomy*: the voice intelligent assistant can work in a fully autonomous mode, without using an Internet connection.

4) *Accuracy*: the assistant correctly recognizes voice input in natural language; fulfills user requests if the required functionality is available or warns about its absence.

Compliance with these requirements is the main criterion for the success of the module development, which should be verified during testing on a specially developed application.

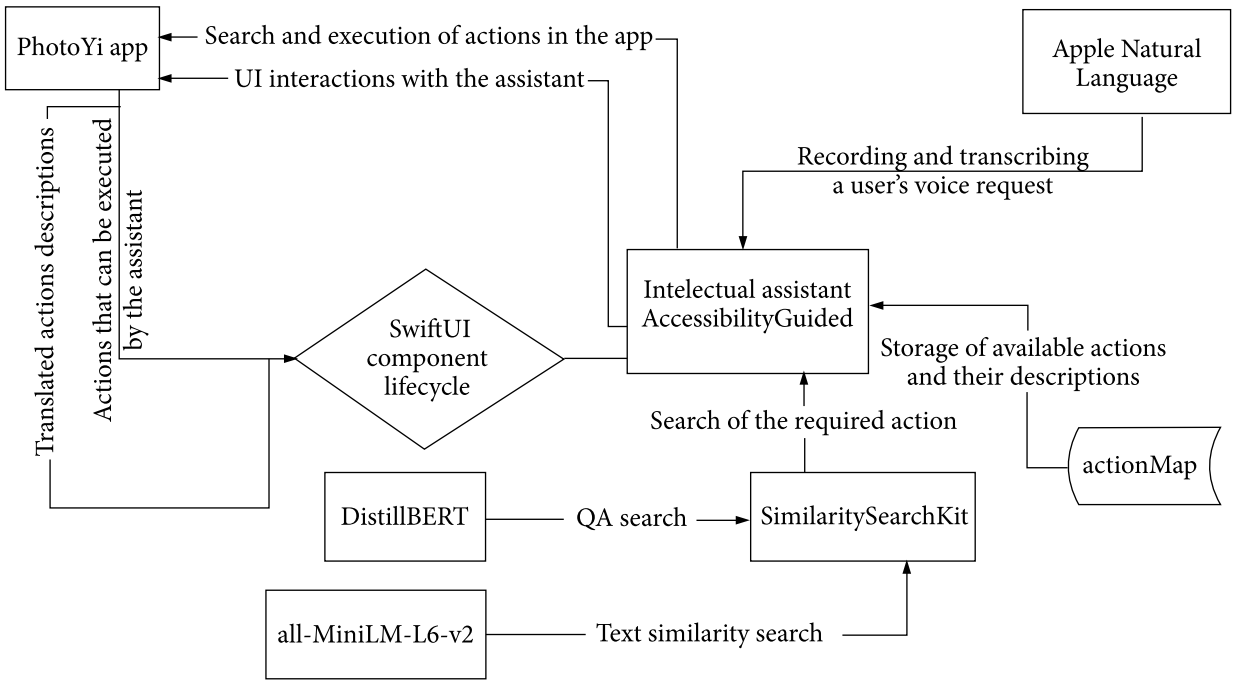

Fig. 1. Structure of the program module

Results

A software module called AccessibilityGuided has been developed in Swift. It is designed to be integrated into iOS applications whose interface is built using the SwiftUI framework, and the minimum supported iOS version is 16.0. This module allows the user to call the desired function within the current window at any time of the application's operation by providing an approximate voice description.

To test the module, a separate photo editor application called PhotoYi was developed to test the assistant's target use cases.

Structure

The developed module for integrating an intelligent assistant consists of several key components (Fig. 1):

1) *User interface*: a visual interface that allows the user to invoke voice input and interact with the intelligent assistant.

2) *Application Programming Interface* (API) for developers, which is used to integrate the module into iOS applications.

3) *Action storage and processing*, which includes methods and structures that store and provide actions available for execution in the current application context.

4) *Speech Recognition*: This includes recording and transcribing the user's voice requests.

5) *Decision-making*, which consists of finding an action that matches the user's request and determining the correctness of the search results.

The proposed structure of the module allows to separate the work with the most important elements: interface operation, AI tools for speech recognition, and AI tools for decision making. This is important in the context of further support and development of the module, as the tools for these elements are rapidly evolving and constantly changing.

Application Programming Interface

The AccessibilityGuided module provides a simple programmatic interface for integration into applications. The integration algorithm is based on linking actions that are relevant for certain application contexts to interface elements. Thus, the actions performed by a certain element, such as

a button, slider, or radio button, are available for execution through the intelligent assistant as long as this element is in the user's field of view.

The module API is represented as 2 methods:

• func guidedAction(descriptionKey: String, action: @escaping() -> Void) — a modifier of a SwiftUI element that accepts an action and a key to a localized text description of this action. The action and its description are added to RAM when the widget becomes visible on the screen and removed when it disappears from the user's view;

• func guidedAction(localizedDescription: String, action: @escaping() -> Void — performs the same function, but instead of the translation key, it accepts a ready-made string with the action description. This modifier is useful when it is necessary to dynamically generate an action description. For example, to allow a parameterized action by creating separate actions and descriptions for each parameter.

These modifiers can be applied to all types of elements in the SwiftUI, allowing the user to perform any action available within the current window through voice requests.

Speech Recognition

The module uses a solution built into iOS — the Speech framework from Apple. This framework has been a part of iOS since version 10.0 and allows transcription of voice input in all languages supported by iOS.

The AccessibilityGuided module's voice input works without an internet connection for a limited list of languages, as it can only work in this mode with those languages for which the Speech framework has the most number of languages for which transcription without an internet connection is possible is support (English, French, etc.). However, with further iOS updates, the growing, so the possibilities for AccessibilityGuided's offline operation will increase with the development of iOS.

Natural Language Processing and Decision Making

All available actions are stored in a dictionary structure along with their descriptions and unique identifiers. To determine the action that matches the user's request, a semantic search based on the Transformer architecture [5] is used, which is a natural language processing method that splits text into vectors of numbers that are compared to classify and search for similar text.

To work with transformer models, the SimilaritySearchKit framework [6] is used, which allows easy import of AI models into iOS and provides ready-to-use models. This module uses 2 transformer models imported from the Hugging Face model library [7]:

• DistilBERT is a lightweight version of the large BERT model from Google [8]. The scalar product is used for the similarity metric between vectors, which allows to correctly identify texts that are similar in meaning, regardless of their size. Initially, it was designed for question-and-answer searches, which is why it is ideal for an intelligent assistant. In this module, this model is used exclusively for English, as it does not support other languages.

• all-MiniLM-L6-v2 — this model belongs to the MiniLM (Mini Language Model) family [9], known for its efficiency and compactness, which makes it ideal for use in environments with limited resources. It is designed primarily to determine the similarity of texts. The similarity metric uses the cosine of the similarity, which is why the size of the texts can affect the result. This model is used in the module for all languages except English.

When searching for the action whose description most closely matches the user's query, the similarity coefficients of the descriptions and the query are compared. The description that has the highest similarity coefficient (which is higher than the threshold set empirically) is considered a result that satisfies the user's query. If none of the similarity coefficients satisfy the similarity threshold, the user is displayed a message about the lack of such a possibility.

Application for Testing the Module

To test the software module, a photo editor application for iOS called Photo*Yi* was created. It creates the necessary conditions for testing the

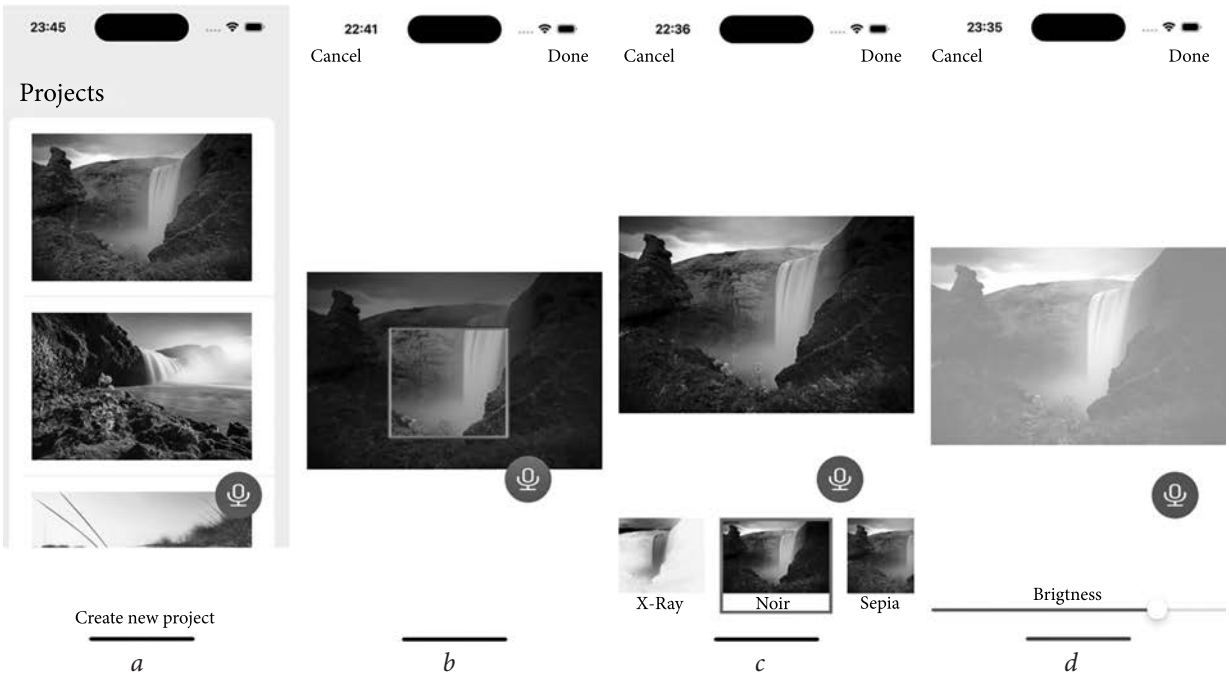

Fig. 2. Interface of the test application (*a*—*d*)

implemented module with an intelligent assistant, as it allows not only to check its functionality but also to analyze the impact on the performance and stability of the application.

The PhotoYi application includes the following image editing functions:

1) Image cropping (Fig. 2, *b*): The user can crop the image by selecting a certain rectangular area on the screen by moving its sides.

2) Apply filters (Fig. 2, *c*): The application offers a set of filters to change the appearance of the image. The filters are located in a long horizontal list and are selected by clicking.

3) Changing the brightness (Fig. 2, *d*): The user can change the brightness of the image by moving the slider.

4) Change the contrast: The application allows you to change the contrast of the image by moving the slider.

5) Change saturation: The user can change the saturation of the image by moving the slider.

In addition, the user can create (Fig. 2, *a*) and delete projects, as well as save and send images.

Photo *Yi* is written in the modern Swift language. The SwiftUI framework was used to develop the interface of the test application, and Core Image was used to implement image editing functions.

Test Results

The photo editor with the integrated module was tested in English and Ukrainian. The testing did not reveal any impact of the assistant on the application's performance or stability.

The primary benchmarks for testing were 4 criteria that were used in the design of the module. It is the compliance with these criteria that forms the assessment of the development results:

1) Universality: the created program module really did not require any preliminary configuration to be integrated into the application. Provided that the developers have written the correct action descriptions, the intelligent assistant performs all the necessary operations.

2) Easy integration: the assistant's program interface consists of 2 methods that can be easily integrated into the existing SwiftUI interface. There is no need to design the integration structure, it is enough to write the correct descriptions of the actions performed by visual elements.

3) Autonomy: for English and several other popular languages, the module works completely autonomously, but for some languages, transcription may require an Internet connection depending on the iOS version.

4) Accuracy: voice input is recognized equally correctly for all languages, while the search for actions that satisfy the query works best for English. For it, the correct results are selected regardless of the size of the texts, while for other languages, the size of the query may affect the quality of the result.

In general, the developed module meets the requirements for ease of implementation and universality, without requiring prior training. The intelligent assistant inside the module provides accurate results using only the power of the mobile device, which meets the requirements of accuracy and autonomy. Thus, the tasks outlined in this article have been fully met.

Conclusion

The developed universal smart assistant integration module demonstrates modern integration capabilities with minimal resource requirements. It is flexible, efficient, and can be easily implemented in a wide range of mobile applications.

It should be noted that the effectiveness of such a solution largely depends on the quality of services provided by developers of artificial intelligence models and speech recognition technologies.

This work has several possible directions for development and research: improving the created software module, as well as defining the architectural approach of the module with development and scaling to other platforms. Taking into account the speed of technology development and the current state of the field of intelligent assistants, the priority research area is the formation of a general architectural approach to the integration of artificial intelligence into arbitrary applications.

REFERENCES

1. Nori, H. et al. (2023). "Capabilities of GPT-4 on medical challenge problems". [online]. Available at: <https://arxiv.org/abs/2303.13375> [Accessed April 21, 2024].
2. Yang, H., Lee, H. (2019). "Understanding user behavior of virtual personal assistant devices". *Inf. Syst. E-Bus Manage*, 17, pp. 65–87. DOI: https://doi.org/10.1007/s10257-018-0375-1.
3. Hoy, M. B. (2018). "Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants". *Medical Reference Services Quarterly*, 37 (1), pp. 81–88. https://doi.org/10.1080/02763869.2018.1404391.
4. Easwara Moorthy, A. and Vu, K.-P.L. (2015). "Privacy Concerns for Use of Voice Activated Personal Assistant in the Public Space", *International Journal of Human-Computer Interaction,* 31(4), pp. 307–335. DOI: https://doi.org/10.1080/10447318.2014.986642.
5. Vaswani, A. et al. (2023). "Attention is all you need", arXiv.org. [online]. Available at: <https://arxiv.org/abs/1706.03762> [Accessed April 28, 2024].
6. ZachNagengast - ZachNagengast/similarity-search-kit. [online]. Available at: <https://github.com/ZachNagengast/similarity-search-kit> [Accessed 11 May 2024].
7. Wolf, Thomas, et al. "HuggingFace's Transformers: State-of-The-Art Natural Language Processing". ArXiv: 1910.03771 [Cs], Feb. 11. 2020, arxiv.org/abs/1910.03771 [Accessed May 13, 2024].
8. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019)." DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". arXiv.org. https://arxiv.org/abs/1910.01108 [Accessed May 13, 2024].
9. Wang, W. *et al.* (2020). "MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". https://arxiv.org/abs/2002.10957 [Accessed June 21, 2024].

Received 28.07.2024

В.Ю. Левчук, магістр,
Національний університет "Києво-Могилянська академія",
вул. Григорія Сковороди, 2, м. Київ, Україна, 04070,
ORCID: https://orcid.org/0009-0001-6613-7478,
pifagor6541@gmail.com

УНІВЕРСАЛЬНИЙ МОДУЛЬ ІНТЕГРАЦІЇ
ІНТЕЛЕКТУАЛЬНОГО ПОМІЧНИКА У ЗАСТОСУНКИ *iOS*

Вступ. Інтеграція інтелектуальних помічників у мобільні додатки стала важливою задачею в сучасному світі, де користувачі дедалі більше використовують технології штучного інтелекту для все більшої кількості задач. Сучасні рішення, такі як *Siri, Google Assistant* і *Amazon Alexa*, пропонують обмежену гнучкість та потребують значних ресурсів для розробки. Це обмежує можливості широкого впровадження їх, особливо у невеликих розробницьких командах. Тому існує потреба у створенні універсального та ефективного модуля для інтеграції інтелектуальних помічників.

Мета статті. Метою цієї роботи є розробка універсального модуля інтеграції інтелектуального помічника, який може бути легко впроваджений у будь-які *iOS*-додатки. Модуль повинен бути автономним, мати мінімальні вимоги до ресурсів і спрощувати процес розробки.

Методи. Проаналізовано сучасні реалізації інтеграції інтелектуальних помічників зі сторонніми додатками та виокремлено головні недоліки. Сформовано критерії до універсальної інтеграції інтелектуального помічника та визначено підхід до інтеграції як розширення доступності. Розроблено програмний модуль, який надає функціонал інтелектуального помічника для інтеграції в додатки *iOS* із мінімальними витратами ресурсів. Використано моделі-трансформери природної мови з бібліотеки *HuggingFace*. Для тестування створено фоторедактор для *iOS*, у який було інтегровано розроблений модуль і який має інтерфейс англійською та українською мовами.

Результати. Розроблено програмний модуль, який дозволяє повне керування застосунком за допомогою голосових запитів та має простий прикладний програмний інтерфейс для інтеграції у довільний *iOS* застосунок. Для інтеграції використано новий підхід, який полягає у прив'язуванні контекстних дій до елементів інтерфейсу та дозволяє інтелектуальному помічнику працювати з усім додатком. Інтелектуальний помічник всередині модуля не залежить від інтернет-з'єднання, не потребує попереднього тренування та забезпечує однакову якість роботи незалежно від сфери діяльності додатка. Найвища якість роботи забезпечується для англійської мови через обмеження використаних моделей-трансформерів природної мови. Реалізований модуль дає змогу знизити затрати на розробку, підвищити продуктивність і забезпечити високу якість взаємодії з користувачем.

Висновки. Розвиток технологій штучного інтелекту та апаратного обладнання сьогодні дає змогу широко використовувати можливості ШІ на обмежених потужностях мобільних пристроїв. Розроблений універсальний модуль інтеграції інтелектуального помічника демонструє сучасні можливості інтеграції з мінімальними потребами у ресурсах. Він є гнучким, ефективним і може бути легко впроваджений у широкий спектр мобільних додатків. Матеріали статті будуть корисними для розробників, які шукають способи покращення функціональності своїх додатків за допомогою інтелектуальних помічників, а також для дослідження покращення структури взаємодії помічників та застосунків.

Ключові слова: *інтелектуальний помічник, штучний інтелект, семантичний пошук, природна мова, модель, машинне навчання, розпізнавання мовлення, графічний інтерфейс.*