УДК 004.5

DOI: 10.18372/2073-4751.75.18013

Гнатюк В.О., к.т.н.,
orcid.org/0000-0002-4916-7149,
Батрак О.Г.,
orcid.org/0000-0002-7983-8118,
Скуратівський А.А.,
Кудренко С.О., к.т.н.,
orcid.org/0000-0002-0759-3908

МЕТОД ОПТИМІЗАЦІЇ РОБОТИ СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ З ВИКОРИСТАННЯМ ВІРТУАЛЬНОГО АСИСТЕНТА НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ

Національний авіаційний університет

viktor.hnatiuk@npp.nau.edu.ua,
olegh.batrak@npp.nau.edu.ua,
7993538@stud.nau.edu.ua,
stanislava@i.ua

Вступ

Актуальність розробки нових методів оптимізації роботи систем масового обслуговування (СМО) в сучасному світі важлива з кількох ключових причин: поліпшення якості обслуговування (ефективніші СМО дозволяють покращити якість обслуговування та задоволеність клієнтів; оптимізація може включати скорочення часу очікування, оптимізацію черг та забезпечення швидкого та ефективного обслуговування), ефективне використання ресурсів (оптимізація СМО дозволяє раціонально використовувати ресурси, такі як робочий час, персонал, обладнання та інфраструктура; це сприяє ефективному використанню коштів та збільшенню продуктивності), використання сучасних технологій (сучасні технології, включаючи штучний інтелект (ШІ), аналіз даних та автоматизацію, відкривають нові можливості для оптимізації СМО; впровадження цих технологій дозволяє створювати більш інтелектуальні та ефективні системи), підвищення конкурентоспроможності бізнесу (оптимізація роботи СМО дозволяє компаніям підвищити конкурентоспроможність на ринку; швидке та якісне обслуговування стає важливим конкурентним перевагою, що залучає більше клієнтів та покращує їхнє сприйняття бренду), відповідь на сучасні виклики та тенденції (з плином часу змінюються вимоги споживачів та технологічні можливості; оптимізація СМО дозволяє відповісти на нові виклики та адаптуватися до змін у споживчих попитах). Отже, розробка нових методів оптимізації СМО є актуальною науковою задачею, оскільки це сприяє покращенню якості обслуговування, більш ефективному використанню ресурсів, використанню сучасних технологій та підвищенню конкурентоспроможності бізнесу.

Мета

Метою роботи є розробка методу оптимізації роботи систем масового обслуговування, з використанням віртуального асистента на базі штучного інтелекту як ефективного інструменту для автоматизації та поліпшення процесів обслуговування користувачів.

Аналіз сучасних наукових досліджень

Сучасні методи оптимізації роботи СМО включають в себе велику кількість технологій, моделей та стратегій. Ось низка наукових досліджень за цією тематикою: автор представляє загальну теорію черг та методи оптимізації в СМО [1], публікація присвячена моделюванню та аналізу систем обслуговування, включаючи розподіл завдань між ресурсами [2], стаття присвячена методам оптимізації розподілу ресурсів в хмарних обчисленнях для

ефективного обслуговування завдань [3], у статті розглядаються методи оцінки довжини черги та керування дозволом на виклики у мережах зі службами з різними характеристиками [4], у статті пропонуються методи машинного навчання для вибору веб-сервісів з урахуванням якості обслуговування [5]. Ці наукові праці представляють деякі з сучасних методів оптимізації СМО, що використовуються в різних галузях, таких як телекомунікації, хмарні обчислення та веб-сервіси.

З огляду на результати аналізу варто зазначити, що сучасні методи оптимізації СМО мають свої переваги, але також існують певні недоліки. Недоліки сучасних методів оптимізації СМО: складність моделювання (багато методів оптимізації вимагають складного математичного моделювання СМО, що може бути складним завданням та вимагати значних обчислювальних ресурсів), чутливість до параметрів (ефективність багатьох методів оптимізації може значно залежати від правильного підбору параметрів моделі, що ускладнює їх практичне застосування та потребує досить точних вхідних даних), обмеженість у реальних умовах (багато моделей базуються на специфічних припущеннях, які можуть не відображати реальні умови СМО, особливо в змінних та непередбачуваних середовищах). Також, варто зазначити переваги СМО з використанням віртуального асистента (ВА) (телеграм бота): автоматизація та ефективність (ВА можуть автоматизувати багато рутинних завдань, полегшуючи роботу співробітників та забезпечуючи ефективне обслуговування клієнтів), покращення якості обслуговування (ВА можуть надати швидку та точну відповідь на запити користувачів, покращуючи загальне враження користувачів від обслуговування), постійна доступність та швидкість відгуку (ВА можуть бути доступні цілодобово та надавати миттєві відповіді, що позитивно впливає на час очікування та задоволення клієнтів), скорочення витрат (використання ВА може допомогти знизити витрати на обслуговування та звільнити людські ресурси для інших важливих завдань), полегшення взаємодії з користувачами (ВА можуть стати зручним інструментом для взаємодії з користувачами, надаючи їм можливість отримати потрібну інформацію та допомогу швидко та легко). Інтеграція ВА у СМО може допомогти максимально використати переваги автоматизації та поліпшити взаємодію з користувачами. Однак важливо розглядати їх як додатковий інструмент, а не як повноцінну заміну людського фактору та експертності.

Розробка методу оптимізації роботи СМО з використанням ВА на базі ШІ

Для розробки ВА необхідно обрати інструменти, платформу та виконаємо наступні етапи.

Етап 1. Реєстрація віртуального асистента.

Для реєстрації ВА необхідно обрати платформу, відповідно до досліджень у роботах [11, 12], оберемо для прикладу систему обміну миттєвими повідомленнями *Telegram*. Для розробки *Telegram*-бота здійснюємо пошук та запуск *BotFather* у *Telegram*. Створюємо нового бота, використовуючи команду */newbot* . Задаємо *name* та *username* бота, *username* повинен бути унікальним, не повторювати існуючі в базі та закінчуватися словом «*bot*». Після створення *Telegram*-бота можемо його налаштувати та редагувати за необхідності, використовуючи меню. Для підключення бота використовується його *API Token*, що є унікальним для кожного *Telegram*-бота.

Етап 2. Розробка логіки бота в Google Apps Script.

При розробці *Telegram*-бота використаємо *Google Apps Script* (*GAS*), що являє собою скриптову платформу, розроблену в *Google* для розробки легких веб додатків на платформі *Google Workspace* (рис. 1). *GAS* початково розроблена Майком Гармом як побічний проєкт під час роботи над *Google Sheets*. Фреймворк базується на *JavaScript* 1.6, але також включає в себе деякі частини з 1.7 та 1.8, а також підмножину *ECMAScript* 5 *API*. Проєкти *GAS* запускуються в інфраструктурі *Google* на

стороні сервера. Згідно *GAS* «забезпечує прості шляхи для автоматизації задач на перетині продуктів *Google* та сторонніх сервісів». *GAS* також являється інструментом для написання розширень для *Google Docs, Sheets* та *Slides*.

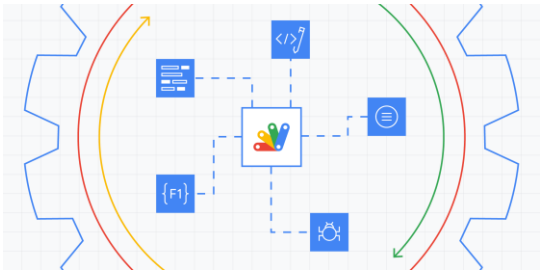

Рис. 1. Скриптова платформа *GAS*

На платформі *GAS* створюємо новий проект, де задаємо змінні ***token*** (токен телеграм-бота, який використовується для зв'язку з *Telegram API*), ***webAppUrl*** (*URL* веб-додатку, який використовується як *webhook* (рис. 2) для отримання вхідних повідомлень від *Telegram, webhook* являє собою метод збільшення або розширення функціональності вебсторінки або вебзастосунку за допомогою користувацьких зворотних викликів (*callbacks*)), ***spreadsheetId*** (ідентифікатор *Google* таблиці, в яку будуть заноситись дані), ***sheetName*** (назва аркуша в *Google* таблиці, в який будуть заноситись повідомлення чату в *Telegram*), ***sheetName2*** (назва аркуша в *Google* таблиці, в який будуть заноситись дані про користувачів).

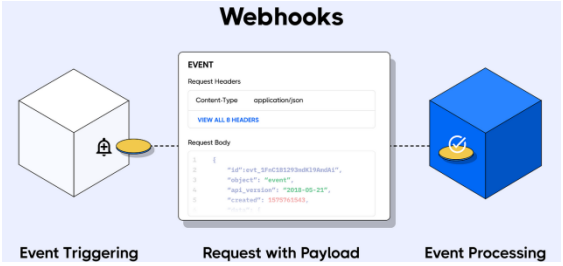

Рис. 2. Метод *webhook*

При розробці *Telegram*-бота використовуємо такі функції: ***setWebhook()*** (встановлює вебхук для телеграм-бота, використовуючи *Telegram API*), ***sendText(chat_id, text, keyBoard, firstName, lastName, currentDate)*** (надсилає текстове повідомлення до користувача за допомогою методу ***sendMessage*** *Telegram API*. Параметри функції включають: ***chat_id*** (ідентифікатор чату), ***text*** (текст повідомлення), ***keyBoard*** (клавіатура для відображення), ***firstName*** (ім'я користувача), ***lastName*** (прізвище користувача) та ***currentDate*** (поточна дата/час)), ***doPost(e)*** (функція обробляє вхідні *HTTP*-запити, що надсилаються веб-додатком, отримує дані про вхідне повідомлення від *Telegram*, розбирає його та заносить необхідні дані до *Google* таблиці).

Також, використовуємо такі об'єкти: ***KEYBOARD_1***, ***KEYBOARD_2*** (об'єкти представляють клавіатури (рис. 3) для відображення у чаті *Telegram*, містять рядки та кнопки, які можна натиснути для взаємодії з ботом).

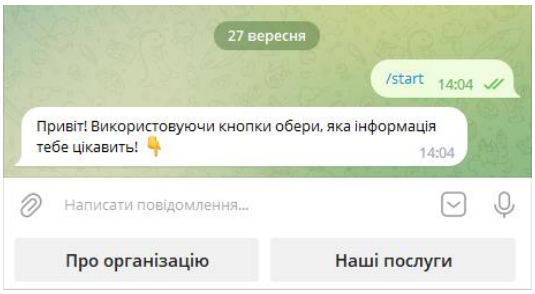

Рис. 3. Клавіатура *Telegram*-бота

Етап 3. Інтеграція ШІ.

Для допомоги користувачам використаємо для прикладу *GPT*-3 (породжувальний попередньо тренований трансформер 3), що являє собою авторегресійну модель мови, яка використовує глибоке навчання, щоби генерувати текст, подібний до людського (рис. 4). Вона є мовною передбачувальною моделлю третього покоління в серії *GPT-n*, створеній *OpenAI*, лабораторією досліджень ШІ в Сан-Франциско [6]. Повна версія *GPT*-3 має ємність у 175 мільярдів параметрів машинного навчання. *GPT*-3, яку було представлено в травні 2020 року і яка перебуває в бета-тестуванні станом на липень 2020 року [7], є частиною тенденції попереднього тренування представлень мови в системах обробки природної мови (ОПМ) [8]. Перед випуском *GPT*-3 найбільшою мовною моделлю була *Turing NLG Microsoft*, представлена в лютому 2020 року, з ємністю в 17

мільярдів параметрів, або менш ніж 10 % у порівнянні з *GPT*-3 [9]. Якість тексту, породжуваного *GPT*-3, є настільки високою, що його складно відрізнити від тексту, написаного людиною, що несе як переваги, так і ризики [9]. Оригінальну працю, яка представила *GPT*-3, презентували тридцять один дослідник та інженер *OpenAI*. У своїй праці вони попередили про небезпеки потенціалу *GPT*-3, й закликали провести дослідження з метою зниження ризику. Девід Чалмерс, австралійський філософ, описав *GPT*-3 як «одну із найцікавіших та найважливіших систем ШІ з будь-коли зроблених» [10].

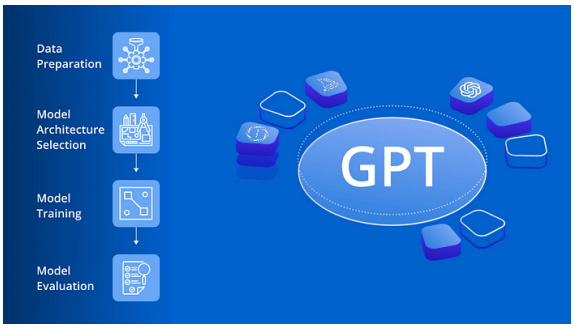

Рис. 4. Модель навчання *GPT*

Для використання *GPT* для допомоги користувачам у телеграм боті та підключення до нього *GAS*, потрібно виконати кілька кроків:

Крок 1. Інтеграція GPT (отримуємо доступ до сервісу, який надає *GPT* (наприклад, *OpenAI GPT*-3) та отримуємо *API* ключ; використовуємо *API* ключ у *GAS* для взаємодії з *GPT*, надсилаючи запити та отримуючи відповіді).

Крок 2. Обробка повідомлень користувачів та відповідей (коли користувач надсилає повідомлення боту у *Telegram*, *GAS* отримує це повідомлення через вебхуки, обробка повідомлення користувача та вилучаємо необхідну інформацію).

Крок 3. Надсилання запитів до GPT та обробка відповідей (складаємо запити до *GPT*, включаючи текст повідомлення користувача або іншу необхідну інформацію; отримуємо відповідь від *GPT* та оброблюємо її для подальшого використання).

Крок 4. Надсилання відповіді користувачеві (створюємо логіку, яка відправляє оброблену відповідь користувачеві у *Telegram* через *Bot API*).

Етап 4. Формування бази даних.

Для формування бази даних буде здійснено запис у *Google Sheets* (*writeToGoogleSheet*) з використанням функції *writeToGoogleSheet*, що відповідає за запис даних користувача та їх запит до *Google Sheets*. Функція додає новий рядок з необхідними даними, у разі помилок вона їх логує. Це дозволяє боту взаємодіяти з користувачем через *Telegram*, за необхідності, викликати ШІ для генерації відповідей та записувати інформацію про користувачів та їх запити до бази даних.

Псевдокод розробленого програмного забезпечення має наступний вигляд (рис. 5).

Функція для встановлення *webhook*:

```
Function setWebhook():
  Try:
    Send POST request to
'https://api.telegram.org/bot{token}/setWe
bhook?url={webAppUrl}'
    Log the response content
  Catch any errors and log them
```

Функція для відправки текстового повідомлення у *Telegram*:

```
Function sendText(chat_id, text,
keyBoard):
  Log 'Sending message with chat_id:',
chat_id, 'text:', text, 'keyboard:', keyBoard
  Create a data object with necessary
parameters for sending a message
  Send a POST request to
'https://api.telegram.org/bot{token}/sendM
essage' with the data
```

Функція для обробки *POST*-запиту:

```
Function doPost(e):
  Try:
    Parse the request contents from e
    Extract chat_id and text from the
message

    If text is "/start":
      Send a welcome message with custom
keyboard
    Else If text is "Text":
      Send a message about Text with
custom keyboard
    Else If text is "Text":
      Send a message about Text with
custom keyboard
```

Else:
Call generateGPT3Response with text and 50 maxTokens
Log GPT-3.5 Response
Send the GPT-3.5 response with custom keyboard

Catch any errors and log them

Функція для виклику *GPT*-3.5 *API*:

Function generateGPT3Response(*prompt, maxTokens*):
Try:
Log 'Calling GPT-3.5 API with prompt:', prompt, 'and maxTokens:', maxTokens

Send a POST request to 'https://api.openai.com/v1/engines/davinci/completions' with the prompt and maxTokens
Add necessary headers including authorization with apiKey and content type

If the response is successful:
Parse the response data and return the trimmed text from GPT-3.5 API
Else:
Throw an error indicating failure to call GPT-3.5 API

Catch any errors and log them

Функція для запису інформації в гугл таблицю:

Function writeToGoogleSheet(*user_id, username, question*):
Try:
Open the Google Sheet associated with the Telegram bot
Append a new row with user_id, username, and question
Close the Google Sheet
Log 'Information successfully written to Google Sheet'
Catch any errors and log them

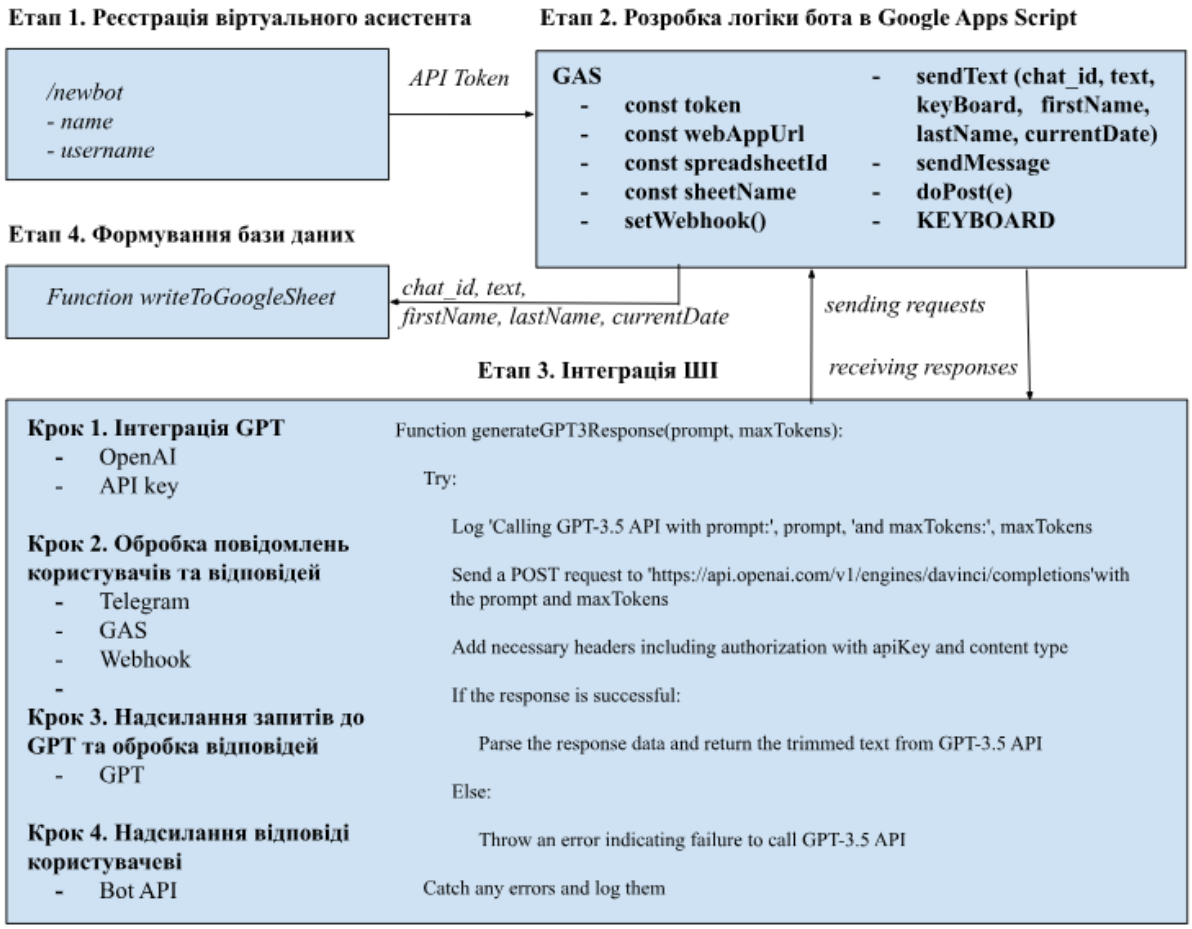

Рис. 5. Схема реалізації
методу оптимізації роботи СМО
з використанням ВА на базі ШІ

У загальному вигляді логіка програмного коду виглядає наступним чином:

1. Встановлення вебхука (*setWebhook*):

- Функція *setWebhook* відправляє *POST*-запит до *Telegram API* для встановлення вебхука, щоб бот отримував повідомлення.
- Успішність встановлення вебхука логується.

2. Відправка повідомлення у *Telegram* (*sendText*):

- Функція *sendText* відправляє текстове повідомлення користувачеві у *Telegram*.
- Вона використовує *POST*-запит до *Telegram API* з необхідними параметрами.

3. Взаємодія з *GPT*-3.5 *API* (*generateGPT3Response*):

- Функція *generateGPT3Response* викликає *API GPT*-3.5, надаючи запит (*prompt*) та максимальну кількість токенів для генерації відповіді.
- Отримана відповідь (текст) від *GPT*-3.5 *API* логується та повертається.

4. Обробка *POST*-запиту (*doPost*):

- Функція *doPost* обробляє *POST*-запит від *Telegram*, що містить дані від користувача.
- Вона розпаковує дані та витягає необхідну інформацію, таку як *chat_id* та текст повідомлення.
- В залежності від тексту, може відправити привітання або запит до *GPT*-3.5 *API*.

5. Запис у *Google Sheets* (*writeToGoogleSheet*):

- Функція *writeToGoogleSheet* відповідає за запис даних користувача та їх запиту до *Google Sheets*.
- Відкриття запиту до *Google Sheets*, додається новий рядок з необхідними даними та закривається запит.
- У разі помилок функція їх логує.

Це загальна логіка програмного коду, що дозволяє боту взаємодіяти з користувачем через *Telegram*, викликати *GPT*-3.5 *API* для генерації відповідей та записувати інформацію про користувачів та їх запити до *Google Sheets* (рис. 6).

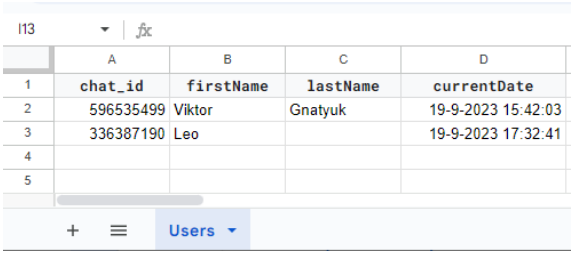

Рис. 6. *Google* таблиця «*Users*»

Таким чином, розроблене рішення для оптимізації роботи СМО з використанням ВА та інтеграції з *GAS*, *Google Tables* та *GPT*. Загальні особливості цього рішення включають наступне: використання ВА, взаємодія з користувачами (бот взаємодіє з користувачами за допомогою кнопок та текстових запитів, дозволяючи їм вибирати інформацію, яка їх цікавить), генерація відповідей за допомогою *GPT*-3.5, збереження інформації в *Google Tables* (інформація про користувачів та їх запити зберігається та управляється в *Google Tables*, спрощуючи роботу з даними та їх аналіз), використання *GAS* для програмування логіки бота та забезпечення інтеграції з *Google Tables*, що дозволяє автоматизувати обробку та збереження даних, оптимізація роботи СМО (ВА та інтеграція з *GPT*-3.5 спрямовані на оптимізацію обслуговування користувачів, забезпечуючи швидку та інформативну відповідь на їх запити).

Це загальні особливості розробленого рішення, яке поєднує в собі ефективну комунікацію з користувачами через ВА, розширення можливостей генерації відповідей за допомогою ШІ, та ефективне управління та аналіз даних через *GAS*.

Висновки

Розроблене рішення розширює можливості існуючих СМО за допомогою використання ВА та розвитку ШІ, зокрема *GPT*-3.5, для забезпечення більш ефективного та інформативного спілкування з користувачами. Таким чином, розроблено ефективний інструмент для автоматизації та поліпшення процесів обслуговування, що забезпечує: покращення взаємодії з

користувачами, використання ШІ для покращення відповідей, ефективне зберігання та аналіз даних, можливість автоматизації завдяки *GAS*. У подальшому планується розширення мовної моделі, удосконалення інтерфейсу користувача, додавання модуля автоматичного розпізнавання мови для підтримки багатьох мов та додаткових можливостей аналізу запитів, розробка алгоритмів, які навчаються відповідям користувачів, щоб надати персоналізовані відповіді та поліпшити досвід взаємодії, дослідження та оптимізація алгоритмів обробки даних для швидшої та ефективнішої роботи системи при великому потоці запитів. Ці наукові розробки можуть покращити ефективність, точність та користувацький досвід СМО з використанням ВА.

Література

1. Kleinrock, L. Queueing Systems, Volume I – Theory. Wiley, 1976. 417 p.

2. Gelenbe E., Mitrani I. Analysis and Synthesis of Computer Systems. New York : Academic Press, 1980. 239 p.

3. Ananthanarayanan, G., et al. CloudScale: Elastic Resource Allocation for Cloud Computing Environments. ACM, 2010.

4. Zhang H., Hou J.C. Queue Length Estimation and Call Admission Control in Differentiated Services Networks. *IEEE/ACM Transactions on Networking*. 2005. Vol. 13. Iss. 2. P. 400–413.

5. Li W., Li Y. Learning Automata-based QoS-aware Web Service Selection. *IEEE Transactions on Services Computing*. 2009. Vol. 2. Iss. 1. P. 48–61.

6. Shead S. Why everyone is talking about the A.I. text generator released by an Elon Musk-backed lab. URL: https://ramaon-healthcare.com/why-everyone-is-talking-about-the-a-i-text-generator-released-by-an-elon-musk-backed-lab/.

7. Bussler F. Will GPT-3 Kill Coding? Towards Data Science. URL: https://towardsdatascience. com/will-gpt-3-kill-coding-630e4518c04d.

8. Brown T.B. et al. Language Models are Few-Shot Learners. URL: https://arxiv.org/abs/2005.14165

9. Sagar R. OpenAI Releases GPT-3, The Largest Model So Far. URL: https://analyticsindiamag.com/open-ai-gpt-3-language-model/.

10. Chalmers D. GPT-3 and General Intelligence. У Weinberg, Justin. Daily Nous. Philosophers On GPT-3 (updated with replies by GPT-3). URL: https://dailynous.com/2020/07/30/philosophers-gpt-3/.

11. Гнатюк В.О., Бондаренко І.О., Каплун І.С. Використання систем обміну миттєвими повідомленнями для автоматизації надання консультативних послуг. *Реєстрація, зберігання і обробка даних*. 2021. Т. 23. № 4. С. 58–67.

12. Гнатюк В.О., Батрак О.Г., Яроцький С.В. Автоматизована система реєстрації місцезнаходження працівника. *Проблеми інформатизації та управління*. 2023. В. 74. № 2. С. 14–20.

Гнатюк В.О., Батрак О.Г., Скуратівський А.А., Кудренко С.О.

МЕТОД ОПТИМІЗАЦІЇ РОБОТИ СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ З ВИКОРИСТАННЯМ ВІРТУАЛЬНОГО АСИСТЕНТА НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ

У науковій статті здійснена розробка методу оптимізації роботи систем масового обслуговування, з використанням віртуального асистента на базі штучного інтелекту як ефективного інструменту для автоматизації та поліпшення процесів обслуговування користувачів. Розроблене рішення забезпечує покращення взаємодії з користувачами, використання штучного інтелекту для покращення відповідей, ефективне зберігання та аналіз даних, можливість автоматизації завдяки GAS. У подальшому планується розширення мовної моделі, удосконалення інтерфейсу користувача, додавання модуля автоматичного розпізнавання мови для підтримки багатьох мов та додаткових можливостей аналізу запитів, розробка алгоритмів, які навчаються відповідям

користувачів, щоб надати персоналізовані відповіді та поліпшити досвід взаємодії, дослідження та оптимізація алгоритмів обробки даних для швидшої та ефективнішої роботи системи при великому потоці запитів. Ці наукові розробки можуть покращити ефективність, точність та користувацький досвід систем масового обслуговування з використанням віртуального асистента.

***Ключові слова**: система масового обслуговування, GAS, віртуальний асистент, штучний асистент, Telegram-бот.*

Gnatyuk V.O., Batrak O.G., Skurativskyi A.A., Kudrenko S.O.

OPTIMIZATION METHOD FOR MASS SERVICE SYSTEM WITH THE USE OF A VIRTUAL ASSISTANT BASED ON ARTIFICIAL INTELLIGENCE

In article, we present the development of an optimization method for mass service systems using a virtual assistant based on artificial intelligence as an effective tool for automating and enhancing user service processes. The proposed solution enhances user interaction, utilizes artificial intelligence to improve responses, efficiently stores and analyzes data, and enables automation through a GAS. Future plans include expanding the language model, improving the user interface, adding automatic language recognition to support multiple languages, and introducing additional query analysis capabilities. We aim to develop algorithms that learn from user responses to provide personalized answers and enhance the user interaction experience. Furthermore, we investigate and optimize data processing algorithms to ensure the system's efficient operation in high query volume scenarios. These research developments have the potential to enhance the efficiency, accuracy, and user experience of mass service systems utilizing a virtual assistant.

***Keywords:** Mass Service System, GAS, Virtual Assistant, Artificial Assistant, Telegram Bot.*