Интеграция с ChatGPT

Сейчас во многие сферы нашей жизни все больше проникает искусственный интеллект. Так, ChatGPT, разработанный OpenAI, является универсальным инструментом, который приносит пользу во многих областях и широко распространен.

Чтобы не отставать от всего мира, мы приняли решение о внедрении возможности работы с ChatGPT на нашей платформе! Это позволит нам расширить ее функциональность и предоставить пользователям доступ к бесконечным знаниям и ресурсам, которые дает этот искусственный интеллект.

На нашей платформе работа с ChatGPT осуществляется при помощи разработанных командой Metabot плагинов.

Плагин для GPT можно найти в **Общих плагинах** на платформе. Он содержит три скрипта:

- **JS скрипт GPT (Common.GPT.GPT)** — вся работа с GPT выполняется через данный скрипт;
- **Session (PHP обертка для V8)** — скрипт **Common.GPT.GPT** сам использует класс сессии, но если необходимо не стандартное поведение, то класс сессии можно использовать напрямую;
- **JS скрипт Session (Common.GPT.Session)** — подключает PHP обертку.

> Рекомендуем ознакомиться с документацией **Плагины**

Как использовать GPT плагин

Есть два способа взаимодействия с GPT в боте:

- Взаимодействие с GPT в команде **"Выполнить JavaScript Callback"**;
- Взаимодействие с GPT в системном скрипте с типом **"Fallback"**.

Первый вариант подходит для случаев, когда GPT используется в боте точечно в определённых скриптах. Этот вариант является предпочтительным и с более управляемой логикой под каждый скрипт. Логика такого скрипта является более понятной и "прямой".

Взаимодействие с GPT в системном скрипте с типом "Fallback" подходит для случаев, когда GPT используется во всех или в большинстве скриптов бота. В этом варианте следует продумывать код так, чтобы он учитывал все возможные кейсы взаимодействия с GPT.

> Желательно**использовать GPT** не во всех скриптах бота, а **только в определённых**, чтобы не загружать GPT большим количеством запросов и не увеличивать время его ответа.

Рассмотрим использование обоих вариантов на примере.

Скрипт "Общаться только с GPT"

В скрипте размещаем две команды:

- "Выполнить JavaScript";
- "Выполнить JavaScript Callback".

Далее создаем меню, которое будет автоматически добавляться к GPT ответам.

В первой команде перед запуском Callback подключаем и инициализируем GPT (1-2 строка), чистим текущую сессию (4 строка) и запоминаем контекст с кодом текущего скрипта (**gpt.setContext(bot.getCurrentScriptCode())** / **gpt.setContext(null)** / **gpt.disableByContext()**). Контекст можно использовать, например, для поиска инструкции Prompt для GPT, или для выключения GPT в Fallback с помощью **gpt.disableByContext()**.

```
const Gpt = require('Common.GPT.GPT')
const gpt = new Gpt()
gpt.setContext(null)
gpt.clearSession()
```

В JavaScript Callback получаем меню из текущего скрипта (2 строка), и добавляем команды для его вывода (5-11 строки). Если пользователь нажал на кнопку, происходит выход из Callback, с помощью планировщика.

Есть два варианта запуска другого скрипта при нажатии на кнопку выхода из меню пользователем:

```
bot.scheduleScript(runScriptId, leadId)
throw new Error('stop') // ИЛИ ДОБАВИТЬ КОМАНДУ СТОП, ПОСЛЕ ДАННОЙ КОМАНДЫ JS-CALLBACK
```

```
bot.run({
    "script_id": runScriptId,
    //"script_code": "your_script_code",
    //"skip_till_command_id": 111,
    //"stop_current_flow": false,
  })
```

```
// ДАННАЯ ТОЧКА КОДА ДОЛЖНЫ ВЫПОЛНЯТЬСЯ, ТК СКРИПТ ПРЕРВАН ВЫШЕ С ПОМОЩЬЮ МОМЕНТАЛЬНОГО ВЫПОЛНЕНИЯ СКРИПТА (bot.runJob)
bot.sendMessage('ОШИБКА 1! Данная точка кода никогда не должна выполняться!')

return true
```

> В скрипте обязательно должен быть прописан один из перечисленных выше вариантов, иначе будет отрисовано лишнее меню из текущей команды.

```
// Получаем меню из текущего скрипта
let menuButtons = bot.getButtonsForCurrentScript();

// Запустится сразу при выполнении данной команды
if (isFirstImmediateCall) {
  bot.sendMessage(
    'Для выхода из данного режима нажмите кнопку "Назад" или напишите "меню" или "бот", подробнее смотрите в инструкции - Как пользоваться ботом',
    menuButtons
  )
  return false
}

let runScriptId = bot.getScriptIdFromButtonsByIncomingMessage(menuButtons)
// Если нажали на кнопку, выходим из колбэка, с помощью планировщика

if (runScriptId !== null) {
  // Запуск другого скрипта из текущей точки
  bot.run({
    "script_id": runScriptId
  })

  // ДАННАЯ ТОЧКА КОДА ДОЛЖНА ВЫПОЛНЯТЬСЯ, ТАК КАК СКРИПТ ПРЕРВАН ВЫШЕ С ПОМОЩЬЮ МОМЕНТАЛЬНОГО ВЫПОЛНЕНИЯ СКРИПТА (bot.runJob)
  bot.sendMessage('ОШИБКА 1! Данная точка кода никогда не должна выполняться!')

  return true
}
// Код ниже запустится, когда лид ответит находясь в текущем скрипте
```

```
// Подключение и инициализация GPT
const Gpt = require('Common.GPT.GPT')
const gpt = new Gpt()

// Взять входящее сообщение из вебхука, отправить в GPT и вывести ответ
gpt.runFromWebhook(null, menuButtons)

return false // не выходим из JS Callback
```

Если же пользователь ответил оставшись в текущем скрипте, то выполняется подключение и инициализация GPT (31-32 строки), а затем входящее сообщение отправляется GPT из вебхука и его ответ выводится пользователю (35 строка).

> **Для Telegram:** ответное сообщение будет обновляться по мере получения ответов от GPT. Если длина ответа от GPT превышает установленный **maxMessageLength** лимит, то ответ будет отправлен несколькими сообщениями.

После отправки сообщения GPT прикрепит к нему кнопки.

Системный скрипт "Fallback"

В данном скрипте взаимодействие с GPT выполняется при помощи команды **Выполнить JavaScript**. JS скрипт проверит контекст и не будет запускать GPT, если он выключен для данного раздела. Если GPT включен, то скрипт запустит его и выведет ответ с прикрепленным к нему последним меню.

Также в Fallback происходит проверка на существования ответа от GPT, и если GPT ответа нет или GPT не запускался в данном разделе, то будет выведен обычный Fallback текст. Для этого в Fallback после команды **Выполнить JavaScript** следует добавить команду **Отправить текст** с условием: **return !(memory.getAttr('has_gpt_answer') * 1)**.

Первым делом в скрипте выполняется подключение и инициализация GPT (2-3 строки). Затем выполняется проверка на включение GPT в разделе из которого был совершен переход в скрипт Fallback (6 строка).

```
// Подключение и инициализация GPT
const Gpt = require('Common.GPT.GPT')
const gpt = new Gpt()

// Если GPT выключен в данном разделе
if (gpt.isDisabledByContext()) {
  return
```

```
}

// Поиск шаблона промта в кастомной таблице по текущему контексту (коду скрипта)
let gptContext = gpt.getContext()
if (gptContext !== null && typeof gptContext === 'string' && gptContext.length > 0) {
  let items = table.find('gpt_prompts', ['prompt'], [['context', '=', gptContext]])
  if (items.length > 0) {
    let item = items[0]
    if (item.prompt !== null && item.prompt.length > 0) {
      let startPrompt =
      `GPT, сейчас ты в роли чат-бота и являешься консультантом в ИТ-компании, отвечай так, будто ты работаешь в этой компании.
      Если ответа в этом тексте нет, то ответь Клиенту: "Извините, но у меня не хватает знаний для ответа на вопрос".
      Можешь дополнять ответ смайликами (emoji).
      `

      if (gptContext === 'our_products_and_services') {
        startPrompt += ` Ответ клиенту обязательно дополни следующей фразой:
         "Чтобы я смог лучше понять, что именно вы хотите, перейдите, пожалуйста, в соответствующий раздел продуктов или услуг на нашем сайте."
        `
      }

      if (gptContext === 'our_products' || gptContext === 'our_services') {
        startPrompt += `Обязательно строй ответы так, чтобы пытаться продать товар.`
      }

      gpt.gptSession.addMessage('system_start', startPrompt)
      gpt.gptSession.addMessage('system', item.prompt)
    }
  }
}

// Взять входящее сообщение из вебхука, отправить в GPT и вывести ответ
gpt.runFromWebhook()
```

Затем происходит поиск шаблона prompt в кастомной таблице по текущему коду скрипта. Если проверка прошла успешно, то GPT отправляются указания для построения ответа в зависимости от контекста (17-31 строки). Последним этапом входящее сообщение

отправляется GPT из вебхука и его ответ выводится пользователю (40 строка).

База знаний и GPT

Существует еще один вариант использования GPT в боте. Для него понадобится создать свою базу знаний, с которой будет работать GPT.

Логика работы: за prompt в данном случае мы берем текст статей. Так как невозможно вместить всю статью в prompt запрос, текст дробится на части и вычисляются эмбеддинги-векторы, эти векторы сохраняются в БД бота. При получении вопроса от клиента, вычисляется его эмбеддинг и ищется его соответствие по статьям. Из текста найденной статьи формируется prompt и полученный ответ отправляется пользователю.

> **Эмбеддинг** — результат процесса преобразования языковой сущности – слова, предложения, параграфа или целого текста в числовой вектор.

Для реализации базы знаний потребуется создать таблицу **gpt_knowledge_base** и скрипт, состоящий из двух команд: **Выполнить JavaScript** и **Выполнить JavaScript Callback** и пунктов меню, которые крепятся к ответу.

Таблица должна содержать следующие поля:

Поле	Тип	Описание
id	AUTOINCREMENT	ID статьи
context	TEXT	Контекст статьи
content	TEXTAREA	Содержимое статьи
embeddings	VECTOR	Эмбеддинг-вектор текста статьи

Далее необходимо добавить в таблицу необходимые "статьи" и заполнить поле embeddings, например, при помощи следующего кода:

```
let tmp = kb.getRows(null, 1)
let article = tmp.length > 0 ? tmp[0] : null
if (article !== null) {
  let embeddings = gpt.gptDriver.getEmbeddings(article['content'])
  kb.saveEmbeddings(article['id'], embeddings)
}
return false
```

В команде **Выполнить JavaScript** в скрипте "База знаний" подключаем и инициализируем GPT (1-2 строка), чистим текущую сессию (4 строка) и запоминаем контекст с кодом

текущего скрипта (**gpt.setContext(bot.getCurrentScriptCode())** / **gpt.setContext(null)** / **gpt.disableByContext()**).

```
const Gpt = require('Common.GPT.GPT')
const gpt = new Gpt()
gpt.setContext(null)
gpt.clearSession()
```

В команде **Выполнить JavaScript Callback** повторно подключаем и инициализируем GPT.

```
// Подключенире и инициализация GPT
const Gpt = require('Common.GPT.GPT')
const gpt = new Gpt()

const callbackStartMessage = 'Для выхода из данного режима нажмите кнопку "Назад" или напишите "меню" или "бот", подробнее смотрите в инструкции – Как пользоваться ботом'
const kbContext = null // контекст поиска статей
const kbLimit = 1 // количество статей передаваемых в prompt
const kbMaxDistance = 0.17 // для отсечки найденных статей по косинусному расстоянию

let prompt = `GPT, сейчас ты в роли чат-бота и являешься консультантом в ИТ-компании, отвечай так, будто ты работаешь в этой компании.
Ответ на вопрос ищи ТОЛЬКО среди информации, которая тебе отправлена, ничего не придумывая и не ищя ответ в своих общих знаниях.
Добавляй emoji в предложения с ответом, там где это уместно.
Если ответ не возможно сформировать на основе предоставленной информации, то ответь Клиенту: "Извините, но у меня не хватает знаний для ответа на вопрос"`

return gpt.runForKnowledgeBase(callbackStartMessage, prompt, kbContext, kbLimit, kbMaxDistance)
```

Затем отправляем запрос к GPT с сформированным заранее prompt (15 строка).

Версия #5
Ирина Петрова создал 16 October 2023 10:40:40
alex обновил 28 August 2024 18:37:08