МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
И.о. заведующего кафедрой
канд. физ.-мат. наук, доц.
_____ А.В. Письменский
_____03. 06_____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

РАЗРАБОТКА ЧАТ–БОТА ДЛЯ DISCORD СЕРВЕРОВ

Работу выполнил _____ П.П.Лагун
(подпись)

Направление подготовки 09.03.03 Прикладная информатика

Направленность Прикладная информатика в экономике

Научный руководитель
канд. физ.-мат. наук, доц. _____ А.В. Письменский
(подпись)

Нормоконтролер
преподаватель _____ Е.В. Горбачева
(подпись)

Краснодар

2024

РЕФЕРАТ

Выпускная квалификационная работа 50 с., 8 ч., 37 рис., 1 табл., 14 источн.

БОТ, ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ, НЕЙРОН, DISCORD, PYTHON

Объектом исследования являются Discord боты с применением искусственных нейронных сетей, предназначенные для оптимизирования и добавления функций в повседневное общение пользователей приложения Discord.

Целью работы является разработать чат-бота для Discord серверов с основным музыкальным направлением, с добавлением полезного для оптимизации и ускорения повседневных задач функционала.

В результате выполнения данной работы рассмотрены теоретические основы ботов, нейронных сетей, применяющиеся методы обучения, сбора и анализа данных, проанализированы перспективы применения нейронных сетей и внедрения Discord ботов и не только в повседневную жизнь.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Технологии с каждым годом просачиваются в жизни людей все больше и больше, позволяя упростить большинство повседневных задач. Одной из подобных технологий являются боты – специальные роботы или искусственный интеллект, которые выполняют автоматизированные задачи по определенному алгоритму, исходя из запроса от пользователя. Они позволяют сократить время пользователя и освободить человеческие ресурсы в виде отдельного рабочего места, которых и так может быть недостаточно или это попросту невыгодно, а также снизить нагрузку на реальных специалистов, уменьшив нецелевые обращения.

Области, в которых боты применяются, безграничны, такие как: техническая поддержка, распространение новостей и информационных сообщений, расчет стоимости товара, оформление заказа, нахождение информации в большом количестве данных, проигрывание музыки, подбор автомобилей, билетов на транспорт, напоминание о важных событиях и т.д.

В большинстве случаев, отличительной чертой работы бота является среда приложения, в которой он применяется, ведь именно в нее внедряется бот и использует интерфейс приложения. Самые известные приложения, в которых используются боты – это социальные сети, мессенджеры, всевозможные сайты, такие как: Telegram, VK, Discord и т.д.

Главной задачей бота является освободить человека от рутинной, однообразной работы и уменьшить время ожидания ответа на запрос клиента, поэтому многие компании внедряют технологии ботов в свои проекты для экономии рабочих мест и в тоже время увеличения клиентоориентированности. Поэтому важно для разработчика правильно настроить работу бота, чтобы он не отпугивал клиента бесконечными уточняющими вопросами, долгими или неправильными ответами. Если это не учитывать в процессе разработки, то при работе бота огромное количество

пользователей попросту закроет приложение и обратится в другой сервис или к конкуренту.

В свою очередь, применение нейронных сетей совместно с ботом обеспечит больший функционал, позволит работать с большим количеством данных и поиском нужной информации, осуществит быстродействие работы приложения.

1 Понятие искусственных нейронных сетей

Нейронные сети (НС) – математические модели, которые построены по аналогии с биологическими нейронными сетями. После разработки специальных алгоритмов обучения нейронные сети находят широкое применение в различных областях, таких как прогнозирование, распознавание образов, управление и другие.

Исследования в области нейронных сетей начались еще в 40-е годы XX века, когда Маккалокком и Питтсом были разработаны первые систематические модели искусственных нейронных сетей.

Работа искусственной нейронной сети основана на взаимодействии простых процессоров (нейронов), которые обмениваются сигналами между собой. Благодаря соединению большого количества таких простых процессоров нейронные сети способны выполнять сложные задачи, которые для человека требовали бы много времени и усилий.

Нейронные сети представляют собой инновационную технологию вычислений, которая открывает новые возможности для исследования динамических задач в финансовой сфере. Начиная с распознавания образов, они быстро развиваются, интегрируя статистические и искусственный интеллект для поддержки принятия решений в финансовой области.

Нейронные сети не программируются в традиционном смысле, а обучаются. Это придает им главное преимущество перед другими алгоритмами. Во время обучения нейронная сеть находит оптимальные коэффициенты связей между нейронами, позволяя выявлять сложные зависимости между входными и выходными данными, а также классифицировать данные согласно их похожести или объединять их. Корректное обучение делает искусственную нейронную сеть способной предсказывать результаты на основе неполных, зашумленных или даже

искаженных данных, что делает их невероятно эффективными для решения сложных финансовых задач.

Искусственную нейронную сеть можно рассмотреть, как направленный граф, который имеет связи между своими узлами, являющимися искусственными нейронами.

По архитектуре связей можно выделить три класса нейронных сетей:

1) однослойные сети прямого распространения (рисунок 1.a);
2) многослойные сети прямого распространения (рисунок 1.b);
3) рекуррентные сети (рисунок 1.c, 1.d).

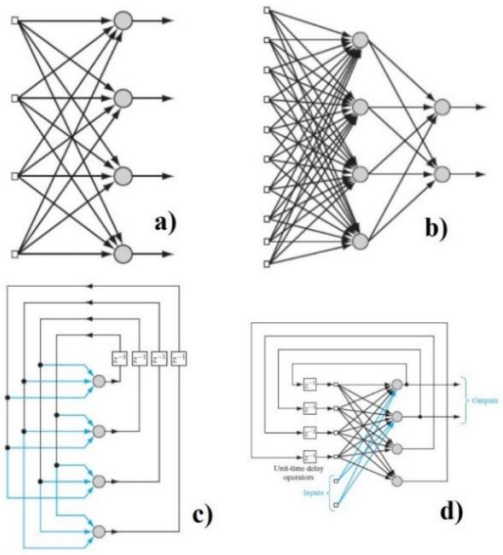


Рисунок 1 – Виды искусственных нейронных сетей

Нейронная сеть организована слоями. В простейшей однослойной форме такой сети имеются входные данные, которые проецируются соответственно на выходной слой нейронов. Такая конструкция имеет строго прямой тип подачи. К слою относятся вычислительные, то есть выходные узлы, но не учитывается входной слой, поскольку там не выполняется никаких вычислений.

Второй класс прямой нейронной сети отличается наличием одного или нескольких скрытых слоев. В данной форме слой входных данных также не учитывается.

В рекуррентных сетях нейронные связи к верхнему слою могут идти не только от нижнего слоя, но и от предыдущего значения самого этого нейрона или других нейронов того же слоя.

1.2 Преимущества и недостатки искусственных нейронных сетей

Применение нейронных сетей для решения различных задач с каждым годом становится популярнее. Создаются гораздо более эффективные программные реализации, построенные по принципу работы нейронных сетей живых организмов. Область применения этой технологии делается еще обширней. Но все же, использование этого подхода имеет свои преимущества и недостатки.

К преимуществам нейронных сетей можно отнести четыре следующих фактора:

1) устойчивость входных данных к шумам. Обученная нейронная сеть способна игнорировать сильно зашумленные данные, подобно тому, как человек может различать лица прохожих, не обращая внимание на посторонние объекты вроде асфальта, одежды, светофоров и пр.

2) адаптация к изменениям. Нейронная сеть способна адаптироваться к изменениям во входных данных, позволяя непрерывно продолжать обучение и работу.

3) отказоустойчивость. Функционирование нейронной сети может продолжаться даже при значительных повреждениях некоторого количества нейронов.

4) быстродействие. Решение задачи с помощью нейронных сетей находится быстрее, чем при использовании обычных алгоритмов.

Искусственные нейронные сети имеют три серьезных недостатка:

1) неточность решения. Нейронные сети не способны выдавать точный ответ, а дают приблизительное решение задачи, которое, возможно, отличается от правильного на некоторую достаточно малую величину.

2) многошаговость решения. В искусственных нейронных сетях каждый нейрон является независимым. Он получает сигнал и, преобразуя его, отправляет на вход следующему нейрону. В зависимости от соседнего нейрона идет изменение синапсов. Таким образом, нейронная сеть не способна решать задачу последовательно.

3) неспособность решать вычислительные задачи. Этот недостаток можно считать следствием двух предыдущих, то есть с помощью нейронных сетей невозможно получить решения задач, требующих последовательных действий и точного ответа. Такими являются, например, арифметические задачи.

Несмотря на некоторые недостатки, разработка искусственных нейронных сетей является перспективным направлением машинного обучения. Ведется активный поиск средств для минимизации этих недостатков и совершенствования работы алгоритмов обучения нейронных сетей.

2 Работа с данными и обучение нейронных сетей

2.1 Сбор и анализ данных

Для задач, решаемых с помощью нейронных сетей необходимо собрать данные, чтобы в дальнейшем начать обучение. Набор сигналов, подаваемых на вход для обучения, представляет собой совокупность значений входных переменных. Эти данные могут быть симулированными, файлами CSV, Excel или JSON, или информацией из базы данных. Это является отдельной задачей при практической реализации нейронной сети, поскольку от формата данных зависит дальнейшая структура проекта.

Достаточно сложным является вопрос о количестве наблюдений, необходимых для обучения нейронной сети. Большинству реальных задач достаточно нескольких сотен или тысяч таких данных, более сложным проектам может потребоваться большее количество.

Входные данные в обучающей выборке не должны быть противоречивыми, иначе это приведет к плохому качеству обучения сети. Также, необходимо, чтобы они содержали истинную информацию о предметной области.

Между данными, подающимися на вход нейронной сети может существовать функциональная или стохастическая связь. Эту зависимость можно выразить в числовой форме. Одним из способов оценки силы такой связи между параметрами является вычисление коэффициента корреляции. Но данная характеристика имеет смысл только для данных, имеющих стохастический тип связи, поскольку при функциональной зависимости параметров коэффициент равен единице.

Для работы с нейронной сетью, в основном, используются данные, которые являются случайными величинами и не имеют функциональной

зависимости, следовательно, вычисление коэффициента корреляции можно считать оптимальным способом оценки возможности применения данных.

Пределы изменения коэффициента находятся на отрезке [-1;1]. Если вычисленный результат отрицательный, то это означает, что с увеличением значения одной переменной значение другой убывает. Если данные независимы, то коэффициент корреляции равен 0, но если результат не равен 0, то между переменными существует зависимость и чем ближе её значение к 1, тем она выше. Силу связи можно оценить, опираясь на классификацию, представленную шкалой Чеддока (таблица 1).

Таблица 1 – Шкала Чеддока силы связей между переменными

Значение	Интерпретация
от 0 до 0.3	слабая
от 0.3 до 0.5	умеренная
от 0.5 до 0.7	заметная
от 0.7 до 0.9	высокая
от 0.9 до 1	очень высокая

Обозначим через $X = (x_1, \dots, x_n)$ – фиксированные значения фактора, $Y = (y_1, \dots, y_n)$ – функции отклика, n – количество наблюдений, тогда для вычисления коэффициента парной корреляции используется формула:

$$r = \frac{\sum(x_i - \bar{x})\sum(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \quad (1)$$

где

$\bar{x}$, $\bar{y}$ – выборочные средние;

$i = \overline{1,n}$.

Для наиболее объективной оценки силы связи между данными, использующимися при разработке модели искусственной нейронной сети, имеет смысл применять коэффициент множественной корреляции, поскольку количество параметров, как правило, больше двух.

Тогда, пусть p – число факторов, $X = \{x_{ij}\}$, где x_{ij} – значение фактора j в исследовании i и $j = \overline{1,p}, Y = (y_1, \dots, y_n)$, где y_i – значение функции отклика в исследовании i, $i = \overline{1,n}$.

Используя введенные обозначения и формулу (1), построим матрицу коэффициентов корреляции:

$$K = \begin{pmatrix} 1 & r_{yx_1} & r_{yx_2} & \cdots & r_{yx_p} \\ r_{x_1y} & 1 & r_{x_1x_2} & \cdots & r_{x_1x_p} \\ \cdots & \cdots & 1 & \cdots & \cdots \\ r_{x_py} & r_{x_px_1} & r_{x_px_2} & \cdots & 1 \end{pmatrix} \quad (2)$$

Используя данную матрицу, можно найти коэффициент множественной корреляции по формуле:

$$R = \sqrt{1 - \frac{D}{D_{11}}} \quad (3)$$

где

D – определитель матрицы K;

D_{11} – определитель K без включения строки и столбца с номером 1.

Таким образом, проанализировав данные с помощью вычисления значений коэффициентов корреляции, можно сделать оценку о возможности дальнейшего применения этого набора параметров для практической реализации нейронной сети.

2.2 Методы обучения

Процесс обучения нейронной сети можно организовать различными способами. Выбор оптимального алгоритма зависит от вида данных и задачи. Существует четыре метода обучения искусственных нейронных сетей.

1) Обучение с учителем предполагает, что набор данных для обучения является размеченным. Это означает, что каждому объекту в обучающем примере соответствует правильный ответ, который и должен получить алгоритм. Как правило, обучение с учителем применяется для решения задач двух типов: регрессии и классификации.

В задачах классификации алгоритм делает предсказание номера класса, к которому принадлежат объекты.

В задачах регрессии данные должны быть непрерывными. Например, при линейной регрессии, ожидаемое значение переменной y вычисляется, учитывая определенные значения x.

Таким образом, обучение с учителем больше всего подходит для решения задач, имеющих внушительную базу данных для обучения.

2) Нередко перед алгоритмом может быть поставлена задача найти заранее неизвестные решения. Для таких задач применяется обучение без учителя, когда у модели есть только набор данных и нет никакой информации по поводу дальнейших действий с ними. В этом случае нейронная сеть самостоятельно пытается проанализировать данные и извлечь из них полезные признаки. Наиболее популярными для данного алгоритма являются задачи кластеризации, обнаружения аномалий и ассоциаций. В задачах кластеризации алгоритм ищет общие признаки при анализе данных и группирует их вместе.

При поиске аномалий, выявляются данные, не соответствующие понятию нормального поведения.

В задачах ассоциаций, алгоритм, рассматривая некоторое количество ключевых признаков, может сделать предсказание о возможной связи одного объекта с другими.

Главным недостатком данного метода обучения является то, что вычислить точность алгоритма достаточно проблематично, поскольку отсутствуют правильные ответы. Но, как правило, размеченные данные затруднительно получить. В таких случаях используют алгоритмы обучения без учителя для поиска зависимостей и решения задачи.

3) В обучении с частичным привлечением учителя используют данные, в которых присутствуют как размеченные, так и неразмеченные объекты.

Этот метод обучения распространен при анализе медицинских изображений, где разметить все объекты является невыполнимой задачей, поскольку появляется вероятность ошибки при постановке правильного диагноза и данные не будут являться правдивыми. В этом случае, нейронная сеть извлекает информацию из размеченных данных и с ее помощью может улучшиться точность предсказаний, по сравнению с алгоритмом обучения, работающем исключительно на неразмеченных данных.

Метод обучения с частичным привлечением учителя полезен, когда проблематично извлечь из данных важные признаки, а разметить все объекты не представляется возможным.

4) Основная идея обучения с подкреплением состоит в принятии определенных действий агентом, взаимодействуя с окружающей средой, за которые он получает поощрение и продолжает совершать какую-либо деятельность, пытаясь увеличить свою награду. Такой подход очень полезен при обучении роботов, которые, например, управляют автономными транспортными средствами.

3 Обучение бота

3.1 Обучение пониманию речи и текста

На этом этапе бот только распознает звук или его визуальные характеристики. Требуется понимать и анализировать эти характеристики так, чтобы в итоге получилось так, будто это сказал человек.

С этим большое количество проблем: запрос от пользователя нужно проанализировать, тем самым убрать шумы или не воспринимать их в учет, отличить друг от друга слова омонимы или похожие по звучанию слова («мыла» и «мыло»), в итоге, исходя из полученных результатов, выбрать наилучший вариант.

Чат-бот использует языковую и акустическую модели. Они предварительно обучаются на огромном объёме данных, накапливают опыт.

Акустическая модель в реальном времени переводит звук в цифровой формат, нарезает на множество микро-отрезков и относит каждый отрезок к определённой части слова. Таких соотношений большое количество: языковая модель выстраивает последовательность, не путая слова омонимы или же похожие по звучанию слова. Она учитывает, с какой частотой входят или стоят рядом друг с другом звуки.

Если искусственный интеллект принял фразу «выведи значения, полученные в икре», языковая модель поймет, что большая вероятность того, что имеется в виду «выведи значения, полученные в игре». Важно, что обе модели зависят от контекста беседы. Если пользователь общается с чат-ботом на сайте магазина автозапчастей, искусственный интеллект поймёт, что в запросе подразумевается какое-то конкретное наименование детали, например, «Поперечный рычаг верхней подвески».

3.2 Работа со смыслом: технология Natural Language Understanding

Запрос клиента переведен в текст, но его желание и смысл пользователя не определены. Искусственный интеллект начинает мыслить, как человек - соотносит запрос с примерами, на которых учился, и находит наиболее подходящие по смыслу. Бот действует классами, сравнивая каждый запрос с одним из них. А классы, в свою очередь, формируются, исходя из сферы применения бота: покупка авиабилетов, подбор автозапчастей, техническая поддержка.

Например, в класс «подобрать автозапчасть» попадают запросы «автозапчасть по VIN коду», «оформление заказа», «гарантия возврата» и нестандартные, вида «подобрать как». С каждым таким этапом происходит обучение бота, в дальнейшем самостоятельно определяя, какие запросы подходят больше всего по смыслу в определенный класс.

3.3 Работа с контекстом

Чат-бот должен самостоятельно определять и учитывать контекст беседы. К примеру, вопрос «как будет “подобрать автозапчасть” по-английски?» можно определить и как подбор детали, и как запрос к переводчику.

Основная технология современных ботов – понимание естественного языка (NLU). Она позволяет искусственному интеллекту понимать пользователей и запускать нужные параметры для обработки запросов. Для этого нужны все технологические решения, связанные с обработкой естественного языка. К ним относится подход к обработке (гибридный, rule-based, статистический), технологии применения чат-ботов в бизнес-процессы компании (локальные или облачные), технологии распознавания и синтеза речи.

Обучение чат-ботов различается по используемой технологии. Наиболее конкурентоспособными на рынке сегодня являются технологии обучения искусственного интеллекта на основе нейросетей, где бот обучается на выборках ответов. Ему предоставляются конкретные запросы от пользователей, и он обучается классифицировать эти запросы по смыслу. Скорость и качество обучения напрямую зависят от эффективности алгоритма, поскольку для достижения высококачественных результатов требуется меньше примеров.

Лингвисты необходимы чат-ботам для создания правил, по которым искусственный интеллект определяет смысл запроса, а не просто ищет ключевые слова. Эти специалисты разрабатывают алгоритмы поведения ботов и обучают их распознавать различные варианты запросов от пользователей. Программист, занимающийся разработкой ботов в компании Just AI, обычно называется лингвист-разработчиком.

Чат-боты несовершенны из-за отсутствия времени на дополнительное обучение, поверхностного планирования задач и недостаточной базы ответов. Большинство ответов чат-бота предварительно заготовлены и обучены заранее. Однако без дальнейшего обучения на примерах ни один искусственный интеллект или нейросеть не сможет идеально определять запросы.

Программисты должны учитывать детали при разработке кода для чат-ботов, чтобы обрабатывать запросы верно и предоставлять нужные данные. Подход к обучению бота должен быть непрерывным, чтобы он мог реагировать на изменения в запросах клиентов и оставаться актуальным.

Чат-боты имеют преимущества перед мобильными приложениями, поскольку они могут быть более удобными для пользователей и повысить коэффициент клиентоориентированности компании. Многие пользователи предпочитают использовать ботов вместо сайтов или других приложений из-за их удобства и быстродействия. Таким образом, чат-боты могут быть эффективным инструментом продаж и обслуживания клиентов.

Создание бота для компании стоит значительно меньше, чем разработка отдельного полноценного приложения. Настройка и отладка приложения, что является важным аспектом для его корректной работы, занимает много времени. Деньги, вложенные в этот процесс, не гарантируют, что после успешной реализации приложение будет интересно клиентам.

Боты позволяют легко определить сценарий коммуникации. Например, для заказа билетов в кино пользователю может быть удобнее написать боту, чем устанавливать отдельное приложение. Боты легко интегрируются в различные платформы, такие как сайты, приложения технической поддержки, что позволяет клиентам общаться с ними через удобный для них способ.

Привлечение аудитории через новые приложения сложнее, чем через уже установленные. Это подтверждается данными из Quantcast. Боты предоставляют удобные способы общения с клиентами и предлагают несколько вариантов выбора подходящего чат-бота для общения.

Как используются боты для бизнеса, для работы с клиентами?

Во время работы с клиентом бот выполняет такие операции:

– оформление;

– ответы на вопросы;

– уточнение;

– заказ;

– техническая поддержка.

Для работы с сотрудниками:

Не менее популярны корпоративные чат-боты, которых используют для оказания услуг сотрудникам фирмы. Это определенные электронные действия, которые предприятие предлагает работникам. Через него можно:

– получать информацию о товаре на складе;

– выполнять заказы недостающего товара;

– уточнять время работы руководства или определенного сотрудника.

Крупные компании предлагают клиентам на выбор несколько разных чат-ботов из социальных сетей. Можно выбрать удобный способ связи.

На данный момент основные функции ботов включают бот-диалог (вопрос-ответ), бот-автоматическая рассылка и бот-меню (нажатие кнопок для получения информации). Эти функции могут быть адаптированы под различные цели и используются для взаимодействия с пользователями.

4 Анализ предметной области и выбор метода решения задачи

Общение между людьми в мессенджерах является неотъемлемой частью жизни почти каждого человека, живущего в мире технологий, развития и прогресса. И как раз внедрение искусственного интеллекта в повседневное общение людей является перспективным направлением применения нейронных сетей, поскольку задача упрощения общения, экономии времени, удобство использования приложений является немаловажным аспектом пользователей. Это техническая поддержка, поиск и проигрывание музыки, создание чатов для общения с адаптированными ролями участников и т.д. Нейронные сети очень хорошо справляются с задачами подобного рода данных, поскольку они способны решать их, основываясь на выявленных скрытых закономерностях.

В данной работе будет рассмотрена теоретическая модель искусственной нейронной сети, способной получать информацию о пользователе и сервере, выводить последние новости, узнавать погоду в городах по всему миру, генерировать изображения по заданному от пользователя запросу, поиск нужной информации в интернете или статьях на Wikipedia; Введение отчетов или любого другого вида данных, поиск и проигрывание музыки, реализация простого калькулятора для несложных математических операций, предоставления всевозможного развлекательного контента. В дальнейшем, будет рассмотрена практическая реализация решения данной задачи.

5 Нейронные сети на Python

За последнее десятилетие наблюдается значительный интерес к нейронным сетям, которые сегодня являются одним из самых популярных методов анализа данных. В различных областях они показывают более качественные результаты, чем другие методы машинного обучения. Благодаря развитию технологий машинного обучения появились готовые библиотеки, содержащие аппаратную часть, созданные архитектуры сетей и программные средства, упрощающие создание сложных нейронных сетей без необходимости писать код.

Одним из основных языков программирования, используемых для создания искусственных нейронных сетей, является Python. Он обладает простотой в использовании и позволяет разрабатывать сложные алгоритмы с минимальными затратами времени и усилий. Существуют многочисленные библиотеки для работы с нейронными сетями на Python, такие как TensorFlow от Google, Theano и Keras.

Перед применением нейронных сетей на практике важно понять возможные проблемы. Необходимо определить архитектуру сети заранее, так как размер и сложность сети могут оказаться чрезмерно большими. Архитектура нейронной сети создает сложные нелинейные разделяющие поверхности в пространстве входов. Чтобы успешно применить нейронные сети, важно предварительно обработать входные данные и определить количество слоев и элементов в них.

При построении классификатора на основе нейронной сети можно выделить три этапа:

1) данные:

а) разработать базу данных из примеров, характеристик для поставленной задачи.

б) разделить весь объем данных на два множества: обучающее и тестовое.

2) предварительная обработка:

а) выбрать систему признаков, характерных для поставленной задачи, и преобразовать данные соответствующим для подачи на вход сети. В результате желательно получить линейно отделимое представление множества образцов.

б) выбрать систему кодирования выходного значения или значений.

3) конструирование, обучение, тестирование и оценка качества сети:

а) выбрать топологию сети: число элементов и структуру связей (входы, слой, выходы).

б) определить функцию активации, которая будет использоваться.

в) определить нужный алгоритм обучения.

Оценить работу нейронной сети в зависимости от её сложности с целью оптимизировать архитектуру.

6 История создания Discord

6.1 Что такое Discord

Если описать Discord простым и понятным языком, то его определение будет выглядеть следующим образом: Discord – это мессенджер, который предназначен для текстового и голосового общения. Он является одним из самых передовых приложений в своем сегменте и доступен для установки на различные операционные системы, такие как Windows, Android и iOS. Пользователям не требуется оплачивать использование Discord. Программа совершенно бесплатна и доступна для скачивания на официальном сайте, все платные функции лишь добавляют обилие кастомизации профиля, дополнительные эмодзи, неограниченное использование звуковой панели с возможностью брать звуки с других серверов, но в самом общении не дает особых преимуществ, по сравнению с пользователем, который пользуется мессенджером бесплатно. Discord поддерживает различное множество языков, что порадует каждого пользователя, не в зависимости от его происхождения и умения разговаривать только на родном языке.

Изначально он создавался, как бесплатное приложение, которое позволяло общаться геймерам между собой с разных уголков мира. Онлайн-игры для многих всегда славились не только способом провести отлично время, но еще и возможностью получить новые знакомства. Пользователи желали получить возможность проводить время в играх вместе со своими знакомыми и друзьями, поэтому появившийся на рынке приложений Discord – стал очень популярным. Игроки стали пользоваться удобным мессенджером как во время игры, так и в повседневной жизни.

Однако сейчас он стал популярен не только среди игроков, но и среди пользователей, не связанных с играми, которые составляют около третьей части всех пользователей. Ежемесячно пользуются Discord около 100

миллионов человек по всему миру, что свидетельствует о его стремительном росте и развитии. Компания активно работает над расширением функционала и намерена удовлетворить потребности широкого круга пользователей, не ограничиваясь только геймерами.

6.2 История создания мессенджера и его развития

В ходе развития приложения наблюдаются множество интересных моментов. Появление популярной платформы кажется совершенно случайным совпадением обстоятельств. Интересно отметить, что одним из основателей Discord был Джейсон Цитрон, страстный игровой фанат. По слухам, ему даже угрожали отчислением из колледжа из-за слишком большого внимания, уделяемого играм вместо учебы.

Джейсон мечтал о создании игр и поэтому усердно изучал программирование. Его увлечение принесло успех – уже в 2008 году его компания разработала первую игру для iPhone и создала платформу OpenFeint, которая позже привлекла внимание компании Gree.

После успешной продажи своей разработки Цитрон основал новую компанию Hammer & Chisel с целью разработки крупных игр для многих пользователей. В скором времени компания создала многопользовательскую онлайн-игру с голосовым и текстовым чатом.

Одним из главных преимуществ этой игры стал широкий функционал общения для игроков, чего не предоставляли другие платформы в то время. Это стало основой идеи о создании новой платформы для общения геймеров.

Постоянное изучение потребностей пользователей, внедрение новых технологий и развитие привели команду к отличному результату. Разработчики считают датой успешного запуска приложения – 13 мая 2015 года, когда чат начали использовать обычные пользователи, а не только команда разработчиков и их знакомые.

6.3 Чем Discord отличается от других мессенджеров

В интернете опубликовано большое количество таблиц, которые наглядно показывают достоинства голосового и текстового мессенджера Дискорд, по сравнению с другими платформами. В примере, который можно увидеть ниже, описан функционал Discord, TeamSpeak, Skype и Ventrilo. Именно эти площадки для коммуникации геймеров были главными конкурентами.

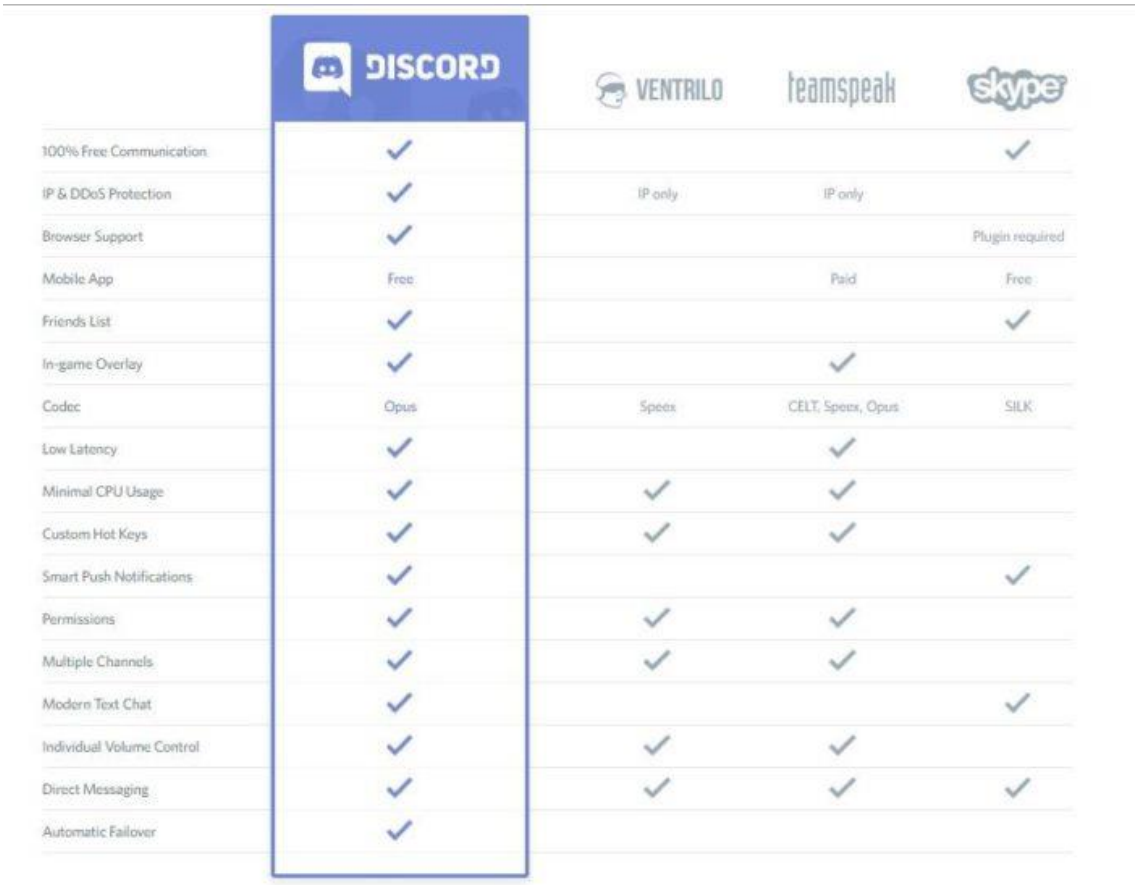


Рисунок 2 – Сравнение мессенджеров

В таблице приведены наглядные сравнения функционала всех ранее описанных приложений. Также пользователи довольно часто еще выделяют важные преимущества Discord. Отнести к таким можно:

– качество звука и возможность его регулировки;

– возможность изменения имени пользователя на каждом сервере;

– функция оверлей;

– возможность отреагировать на сообщения пользователя с помощью эмодзи;

– тонкие настройки;

– наличие меток;

– отображение статуса пользователя.

6.4 В чем секрет успеха Discord

В целом, если перечислить основные достоинства приложения, можно выделить следующие пункты:

– надежность и удобство пользования:

Интуитивный интерфейс приложения позволяет даже новичку легко освоить его. Функционал приложения составлен так, что сообщения в Discord передаются сразу и без задержек, а голосовой чат работает без проблем.

– коммуникация с пользователями:

Команда разработчиков Discord полностью понимает потребности геймеров и сделала всё возможное, чтобы удовлетворить их. Платформа создавалась с учётом всех требований пользователей благодаря постоянной обратной связи с аудиторией.

– коллаборации:

Discord постоянно развивается, сотрудничая с другими крупными игровыми компаниями.

– бесплатный и платный функционал в балансе:

Введение подписок на платформе не повлияло на возможности бесплатного пользования, оставляя пользователям возможность продолжать пользоваться всеми основными функциями. То есть никак не затрагивается возможность общаться в текстовых и голосовых чатах, обмениваться

изображениями или видео, создавать свои сервера и настраивать их так, как нужно.

– наличие реферальной программы:

Большинство пользователей Discord начали использовать платформу по приглашениям, что подтверждает значимость рекомендаций и удобства функции для пользователей.

Популярность мессенджера Discord полностью заслужена, благодаря его сложной истории создания, которую разработчикам удалось успешно преодолеть, предложив уникальное решение, которым пользуются сотни миллионов людей по всему миру. Discord продолжает развиваться, становясь популярным не только среди геймеров, но и у широкой массы пользователей благодаря удобству использования приложения.

7 Примеры работы Discord ботов

Рассмотрим одни из самых популярных примеров работы Discord ботов:

– настройка приветствия и прощания с участником с выдачей роли:

Данная функция позволяет автоматизировать приглашение и выдачу роли новым участникам сервера, что намного облегчает этот процесс и экономит время для создателя сервера.

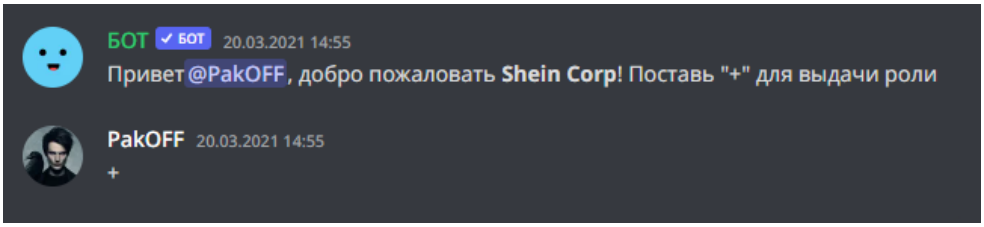


Рисунок 3 – Пример приветствия и выдачи роли

– ведение отчетности с тегом роли ответственного участника, исходя из заполненной Google формы:

Данная функция позволяет вести отчетность по разному виду информации, предварительно полученной из заполненной пользователем Google формы. Здесь реализован тег ответственной за ведение отчетности роли, что, в свою очередь, осуществляет заметность данного сообщения в виде отдельного уведомления с большим приоритетом.

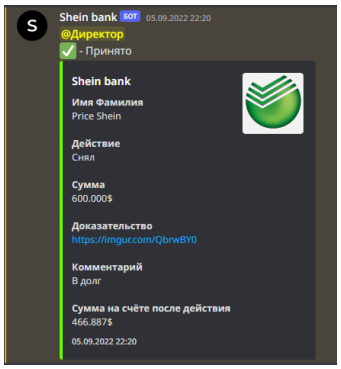


Рисунок 4 – Пример введения отчетности с тегом роли ответственного участника

– поиск и проигрывание музыки:

Данная функция позволяет найти нужный пользователю трек только по названию композиции, что облегчает и ускоряет процесс поиска. После успешного нахождения трека, бот автоматически добавляется в голосовой канал, где уже находится пользователь и начинает воспроизведение найденной композиции.

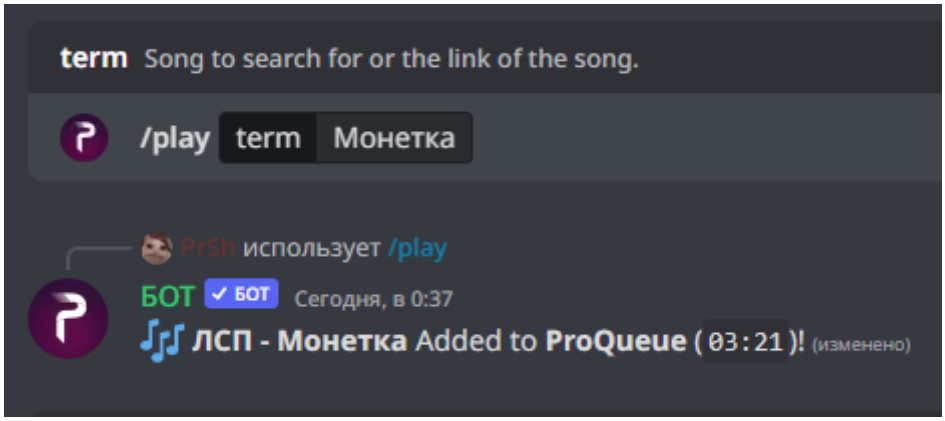


Рисунок 5 – Пример поиска трека по названию и последующего проигрывания

– генерирование случайного числа в заданном диапазоне:

Данная функция позволяет сгенировать случайное число в заданном пользователем диапазоне от 1 до n, где n – число, введенное пользователем.

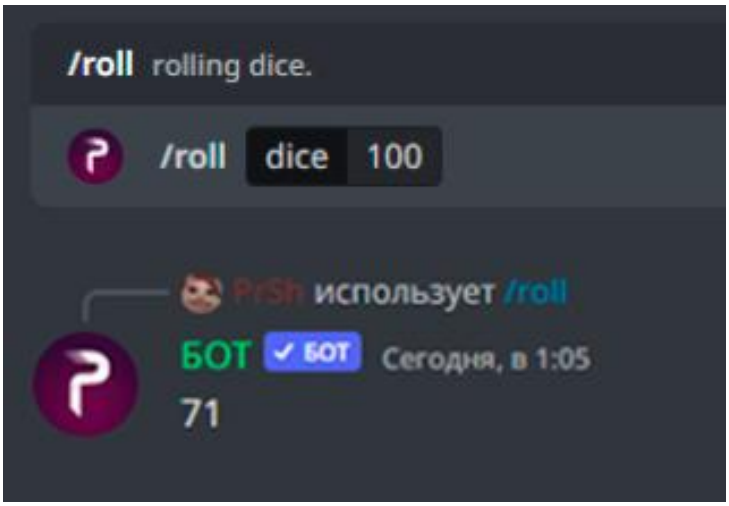


Рисунок 6 – Пример генерации случайного числа, исходя из заданного пользователем диапазона

– получение информации об участнике сервера и о самом сервере:

Данная функция позволяет получить информацию о любом участнике сервера. На рисунке 7 пример информации о профиле конкретного участника сервера, а именно его уровень, репутацию и кредиты, ранг и количество опыта на данный момент. Настройка вывода определенной информации об участнике прерогатива разработчика.

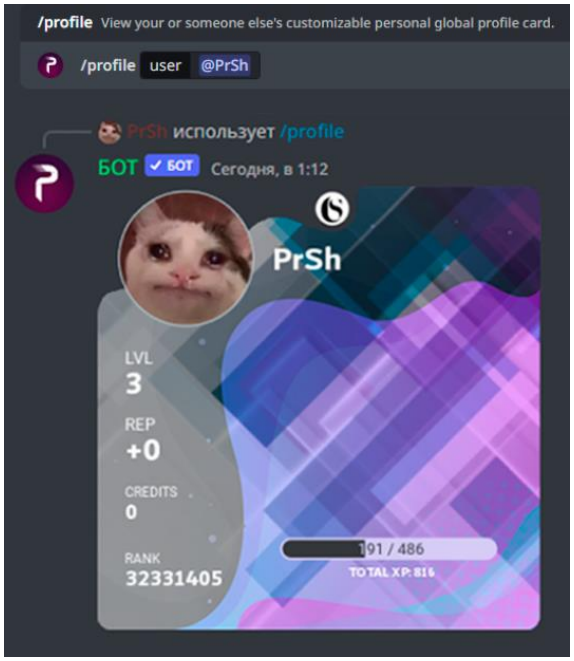


Рисунок 7 – Пример информации об участнике сервера

На рисунке 8 уже вывод информации в целом о пользователе Discord, а именно идентификатор, дата создания учетной записи Discord и присоединения на сервер.

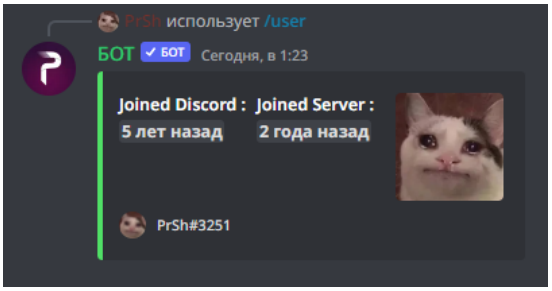


Рисунок 8 – Пример информации о пользователе

На рисунке 9 виден вывод информации о созданном сервере, а именно ID сервера, дата создания, кем был создан, количество участников, количество текстовых и голосовых каналов, количество ролей.

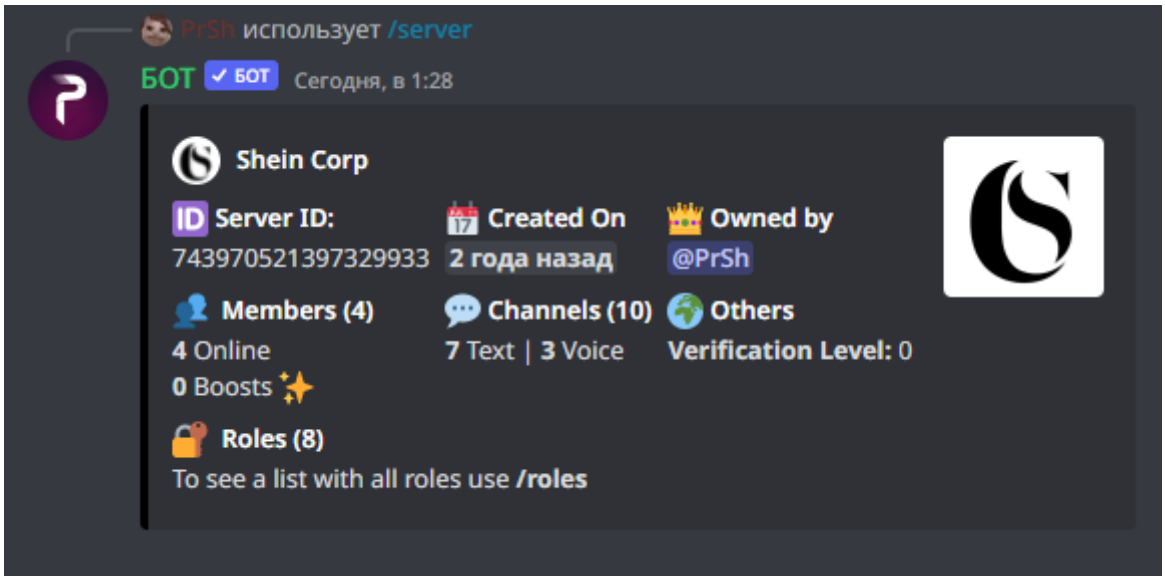


Рисунок 9 – Пример информации о сервере

8 Практическая часть

Перед написанием кода для Discord бота сначала нужно его создать. В этом нам помогает официальный сервис Discord Developer Portal, в котором мы можем настроить имя бота и изображение профиля с настройкой описаний, получить так называемый Token для дальнейшей идентификации бота в нашем коде и подключении его к серверам, настроить его права, функционал и возможности.

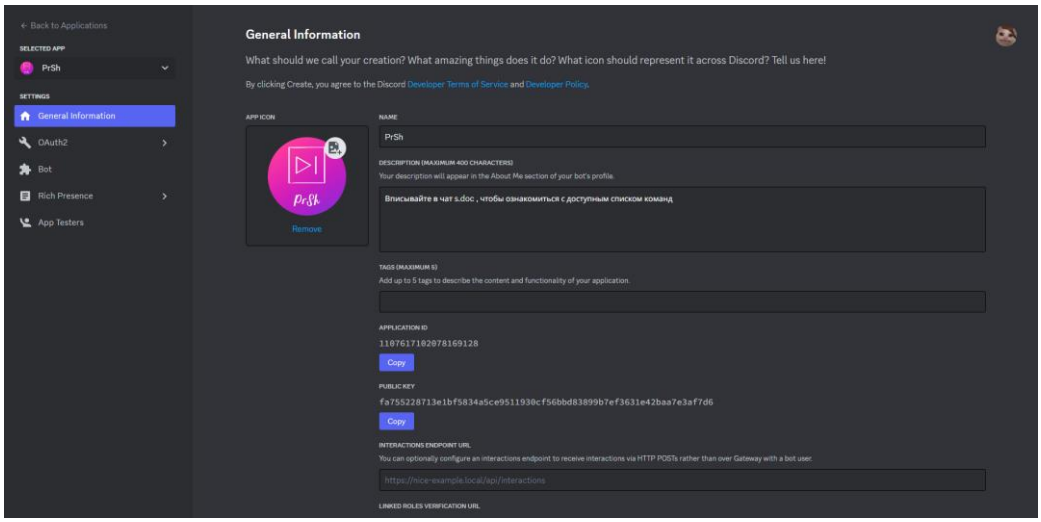


Рисунок 10 – Пример созданного аккаунта бота

Рассмотрим получение Token’а для дальнейшего взаимодействия с ботом и подключения его к серверам на рисунке 11:

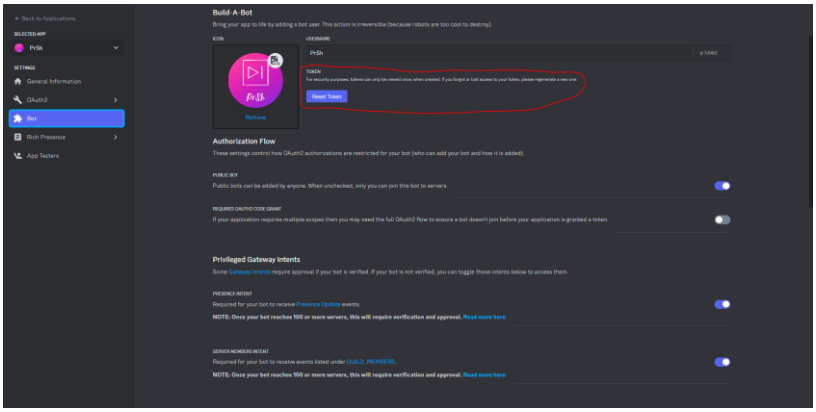


Рисунок 11 – Пример получения Token’а

А также предоставление прав, функционала и возможностей боту на рисунке 12:

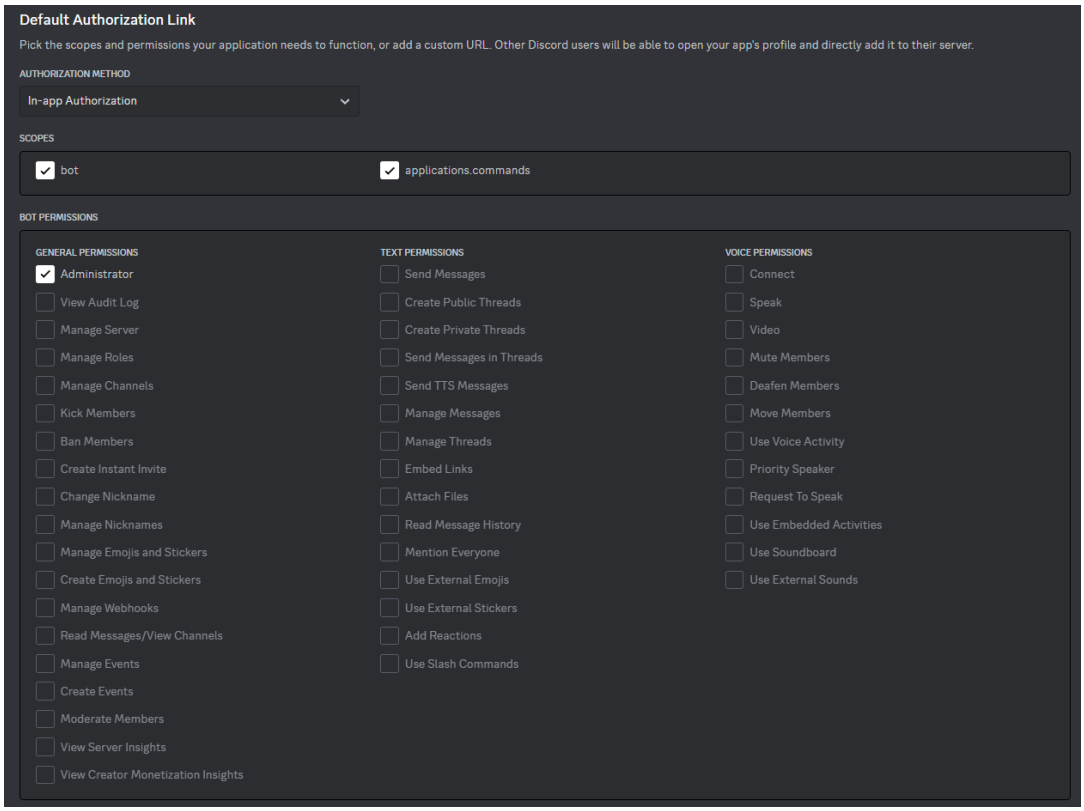


Рисунок 12 – Пример предоставления прав и функционала боту

При выборе роли Administrator бот получает все права и обладает возможностями подобно создателю сервера. При разработке бота я выбрал именно роль Administrator, т.к. бот в первую очередь был создан для применения на личных серверах. В дальнейшем уже при продвижении своего проекта рекомендуется выбирать конкретные права и возможности бота для гарантии безопасности, ведь если Token бота попадет в плохие руки, то взломщик получит все права от сервера, на котором находится бот с ролью Administrator.

После выполнения первых шагов получим два основных элемента:

1) аккаунт бота;
2) Token для подключения бота.

Собственно, вот и всё. На данном этапе наш бот полностью пассивен.

Затем мы переходим уже непосредственно к написанию кода программы, создав предварительно файл main.py, первым делом нам стоит подключить огромное количество библиотек с последующей загрузкой их через командую строку и добавлением переменных в PATH системы, для дальнейшей их успешной работы. В нашем случае при выборе языка программирования Python основной библиотекой является discord.py. Все остальные библиотеки подключаются по мере надобности и добавления функционала боту. Пример подключенных библиотек можно увидеть на рисунке 13:

```python
import discord
from discord.ext import commands, tasks
from discord.ui import Button, View
import random
from discord.ext.commands import Bot
from dotenv import load_dotenv
import os
from requests import get
import openai
import wikipediaapi
from discord.ext.commands import MemberConverter
import re
import uuid
import json
import requests
import yt_dlp
from discord.voice_client import VoiceClient
from pydub import AudioSegment
from pytube import YouTube
from ytmusicapi import YTMusic
import asyncio
import logging
import glob

from discord import Color
from discord.ext.commands import BucketType
import aiohttp
import io
from bs4 import BeautifulSoup
from datetime import datetime
import pandas as pd
# from googleapiclient.discovery import build
from sympy import symbols, solve, Eq
import asyncpg
import ffmpeg
from upstash_redis import Redis
import redis
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
```

Рисунок 13 – Пример подключения библиотек

Также для взаимодействия с ботом и подключения его на сервера, нам нужно вставить в программу Token и префикс для взаимодействия, в моем случае префиксы поделились на две категории, a. для музыки и s. для всех остальных команд. При реализации своего бота я добавил взаимодействие с популярными сервисами искусственного интеллекта OpenAI, Google Api, Search Engine, Hugging Face Api которые в дальнейшем поможет получать информацию для некоторых команд. Пример вставки в код видно на рисунке 14:

```
discordtoken = os.getenv("TOKEN")
openai_api_key = os.getenv("API_KEY")
openai.api_key = openai_api_key
news_key = os.getenv("NEWS_API_KEY")
weather_key = os.getenv("OPEN_WEATHER_KEY")
SEARCH_ENGINE_ID = os.getenv("SEARCH_ENGINE_ID")
API_KEY = os.getenv("GOOGLE_API_KEY")
HUGGING_FACE_API_TOKEN = os.getenv('HUGGING_FACE_API')
```

Рисунок 14 – Пример вставки Token'а и подключение сервиса OpenAI

После всех вышеописанных шагов приступаем к запуску бота и подключения его к серверу с помощью функций @bot.event и on_ready(), с возможностью записи всех ошибок в отдельный файл discordBot.errors, продемонстрировано на рисунках 15 и 16:

```
@bot.event
async def on_ready() -> None:
    logging.basicConfig(filename='discordBot.errors', level=logging.ERROR)
    print(f"{bot.user.name} успешно запущен!")
    print("-"*100)
```

Рисунок 15 – Пример запуска бота

```
C:\WINDOWS\system32>cd C:\Users\User\Desktop\pybot\DiscordBot

C:\Users\User\Desktop\pybot\DiscordBot>python main.py
[2023-05-21 15:40:31] [INFO    ] discord.client: logging in using static token
[2023-05-21 15:40:32] [INFO    ] discord.gateway: Shard ID None has connected to Gateway (Session ID: 227cd6dfeda26d199b021774c4aeb66b).
PrSh успешно запущен!
----------------------------------------------------------------------------------------------------
```

Рисунок 16 – Успешный запуск бота

Также одна из функций бота является уникальное приветствие к нескольким личным серверам. Бот проверяет, к какому именно серверу он подключен и от этого выбирает уникальное приветствие для новоприбывшего пользователя. Видно, это на рисунках 17 и 18:

```python
@bot.event
async def on_member_join(member):
    welcome_channel = discord.utils.get(member.guild.channels, name = "┆🤠┆ну-здарова")
    welcome_channel1 = discord.utils.get(member.guild.channels, name = "выдача-роли")
    welcome_channel2 = discord.utils.get(member.guild.channels, name = "буквы_и_цифры")
    welcome_message = f"Эй, йоу {member.mention} че каво? Ты прибыл на сервер {member.guild}! Ниже есть канал #┆★┆выбор-роли"
    welcome_message1 = f"Добро пожаловать {member.mention} на сервер {member.guild}! Мы рады видеть Вас здесь."
    welcome_message2 = f"Че с деньгами {member.mention}? Ты залетел на сервер {member.guild} и теперь должен косарь!"

    if welcome_channel is not None:
        await welcome_channel.send(welcome_message)
    if welcome_channel1 is not None:
        await welcome_channel1.send(welcome_message1)
    if welcome_channel2 is not None:
        await welcome_channel2.send(welcome_message2)
```

Рисунок 17 – Код уникальных приветствий

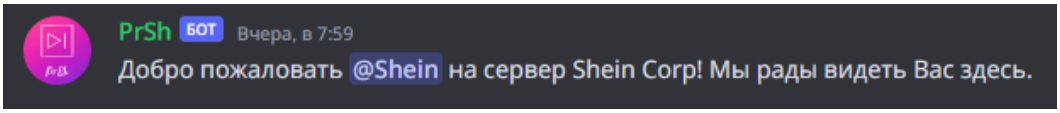
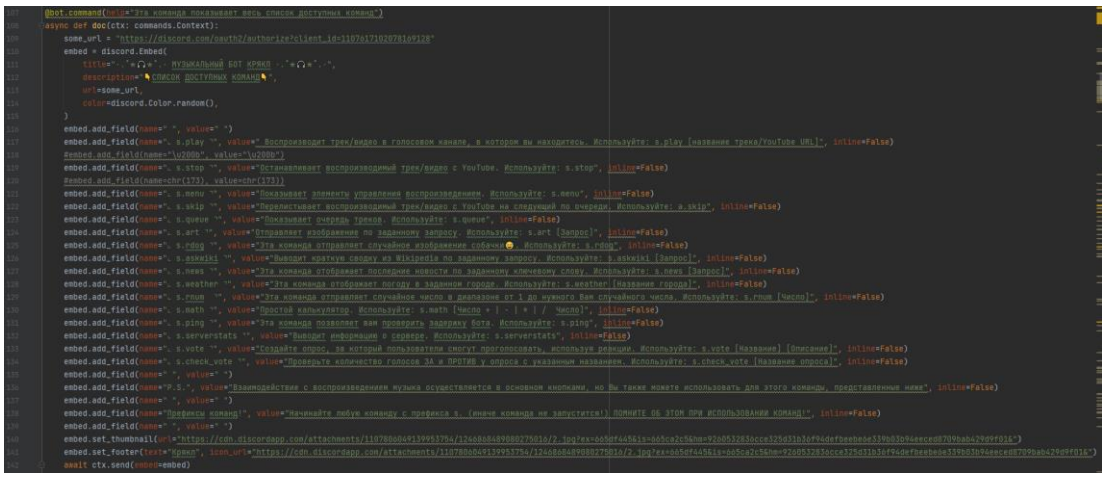


Рисунок 18 – Результат работы кода уникальных приветствий

Переходим уже непосредственно к написанию функциональных команд, с которыми каждый пользователь сервера, на который подключен бот, сможет взаимодействовать. Одна из основных команд является ознакомление со всем списком доступных команд пользователя, ведь без них невозможно взаимодействие с ботом. Добавлена возможность перейти по кликабельному титулу, где хранится ссылка на подключение бота к серверу. Также в правом верхнем углу выведена аватар бота и в нижнем левом углу подпись бота. Реализация показана на рисунках 19 и 20:



```
@bot.command(help="Эта команда показывает весь список доступных команд")
async def doc(ctx: commands.Context):
    some_url = "https://discord.com/oauth2/authorize?client_id=1107617102078169128"
    embed = discord.Embed(
        title="*.'★♫★'.* МУЗЫКАЛЬНЫЙ БОТ КРЯКЛ *.'★♫★'.*",
        description="👇СПИСОК ДОСТУПНЫХ КОМАНД👇",
        url=some_url,
        color=discord.Color.random(),
    )
    embed.add_field(name=" ", value=" ")
    embed.add_field(name="ㄴ s.play ", value=" Воспроизводит трек/видео в голосовом канале, в котором вы находитесь. Используйте: s.play [название трека/YouTube URL]", inline=False)
    #embed.add_field(name="\u200b", value="\u200b")
    embed.add_field(name="ㄴ s.stop ", value="Останавливает воспроизводимый трек/видео с YouTube. Используйте: s.stop", inline=False)
    #embed.add_field(name=chr(173), value=chr(173))
    embed.add_field(name="ㄴ s.menu ", value="Показывает элементы управления воспроизведением. Используйте: s.menu", inline=False)
    embed.add_field(name="ㄴ s.skip ", value="Перелистывает воспроизводимый трек/видео с YouTube на следующий по очереди. Используйте: a.skip", inline=False)
    embed.add_field(name="ㄴ s.queue ", value="Показывает очередь треков. Используйте: s.queue", inline=False)
    embed.add_field(name="ㄴ s.art ", value='Отправляет изображение по заданному запросу. Используйте: s.art [Запрос]', inline=False)
    embed.add_field(name="ㄴ s.rdog ", value="Эта команда отправляет случайное изображение собачки🐶. Используйте: s.rdog", inline=False)
    embed.add_field(name="ㄴ s.askwiki ", value="Выводит краткую сводку из Wikipedia по заданному запросу. Используйте: s.askwiki [Запрос]", inline=False)
    embed.add_field(name="ㄴ s.news ", value="Эта команда отображает последние новости по заданному ключевому слову. Используйте: s.news [Запрос]", inline=False)
    embed.add_field(name="ㄴ s.weather ", value="Эта команда отображает погоду в заданном городе. Используйте: s.weather [Название города]", inline=False)
    embed.add_field(name="ㄴ s.rnum  ", value="Эта команда отправляет случайное число в диапазоне от 1 до нужного Вам случайного числа. Используйте: s.rnum [Число]", inline=False)
    embed.add_field(name="ㄴ s.math ", value="Простой калькулятор. Используйте: s.math [Число + | - | * | / Число]", inline=False)
    embed.add_field(name="ㄴ s.ping ", value="Эта команда позволяет вам проверить задержку бота. Используйте: s.ping", inline=False)
    embed.add_field(name="ㄴ s.serverstats ", value="Выводит информацию о сервере. Используйте: s.serverstats", inline=False)
    embed.add_field(name="ㄴ s.vote ", value="Создайте опрос, за который пользователи смогут проголосовать, используя реакции. Используйте: s.vote [Название] [Описание]", inline=False)
    embed.add_field(name="ㄴ s.check_vote ", value="Проверьте количество голосов ЗА и ПРОТИВ у опроса с указанным названием. Используйте: s.check_vote [Название опроса]", inline=False)
    embed.add_field(name=" ", value=" ")
    embed.add_field(name="P.S.", value="Взаимодействие с воспроизведением музыка осуществляется в основном кнопками, но Вы также можете использовать для этого команды, представленные ниже", inline=False)
    embed.add_field(name=" ", value=" ")
    embed.add_field(name="Префиксы команд!", value="Начинайте любую команду с префикса s. (иначе команда не запустится!) ПОМНИТЕ ОБ ЭТОМ ПРИ ИСПОЛЬЗОВАНИИ КОМАНД!", inline=False)
    embed.add_field(name=" ", value=" ")
    embed.set_thumbnail(url="https://cdn.discordapp.com/attachments/1107880849139953794/1246080849080279016/2.jpg?ex=665bf445&is=665ca2c5&hm=926053283ccce325d31b36f94defbeebebe6339b83b94eeced8709bab429d9f01&")
    embed.set_footer(text="Крякл", icon_url="https://cdn.discordapp.com/attachments/1107880849139953794/1246080849080279016/2.jpg?ex=665bf445&is=665ca2c5&hm=926053283ccce325d31b36f94defbeebebe6339b83b94eeced8709bab429d9f01&")
    await ctx.send(embed=embed)
```

Рисунок 19 – Код списка доступных команд

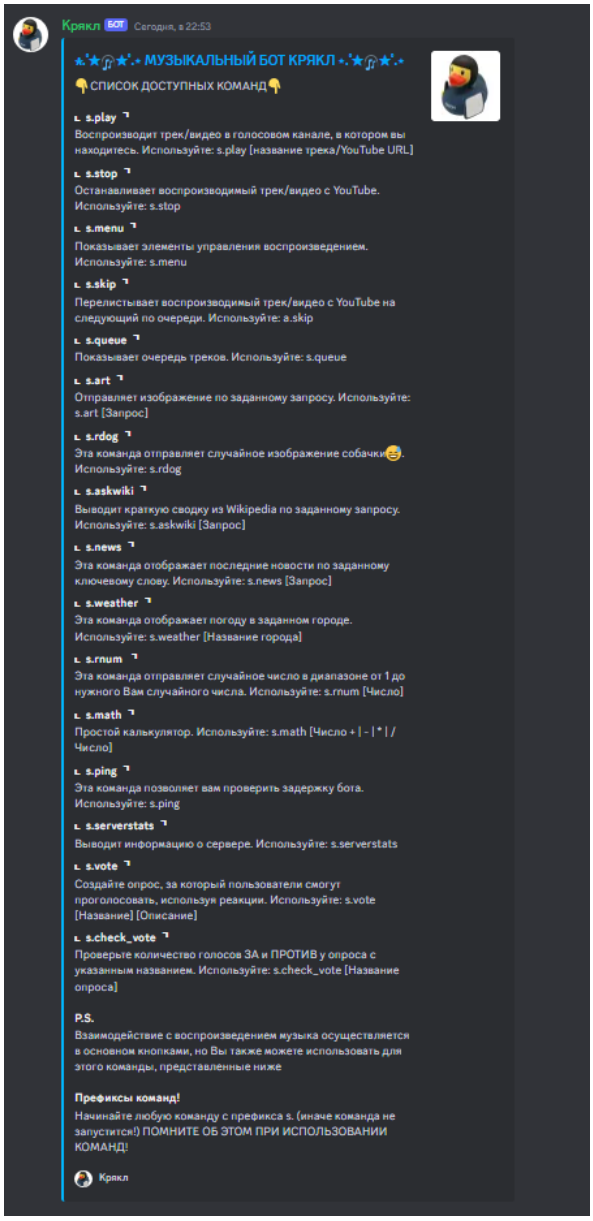


Рисунок 20 – Результат работы кода доступных команд

Одним из основных направлений при разработке Discord бота – проигрывание музыки, ведь по личному опыту именно за этим функционалом я раньше добавлял на свои сервера Discord ботов. Реализация происходила с помощью удобного и популярного сервиса видео хостинга YouTube, что видно на рисунке 21, а результат работы на рисунке 22 и 23:

```
2 usages (1 dynamic)
@bot.command(name='play', help='Воспроизведение песни или плейлиста с YouTube')
async def play(ctx, *, url=""):
    if url:
        await process_playback(ctx, url)

1 usage
@bot.command(name='queue', help='Отображает следующие песни в очереди')
async def show_queue(ctx):
    if queue.empty():
        await ctx.send("Очередь пуста.")
    else:
        response = [f"{i+1}. {YouTube(track.url).title}" for i, track in enumerate(queue._queue)]
        await ctx.send("\n".join(response))

@bot.command(name='skip', help='Пропускает текущий трек')
async def skip(ctx):
    if ctx.voice_client and ctx.voice_client.is_playing():
        ctx.voice_client.stop()
        await asyncio.sleep(1)
        await play_next(ctx)

@bot.command(name='stop', help='Останавливает воспроизведение и очищает очередь')
async def stop(ctx):

    global should_continue_adding,pause
    pause=True
    should_continue_adding = False
    if ctx.voice_client and (ctx.voice_client.is_playing() or ctx.voice_client.is_paused()):
        ctx.voice_client.stop()
        await asyncio.sleep(1)
        for mp4_file in glob.glob('*.mp4'):
            os.remove(mp4_file)
    while not queue.empty():
        queue.get_nowait()
    await skip(ctx)
```

Рисунок 21 – Фрагмент кода реализации проигрывания музыки

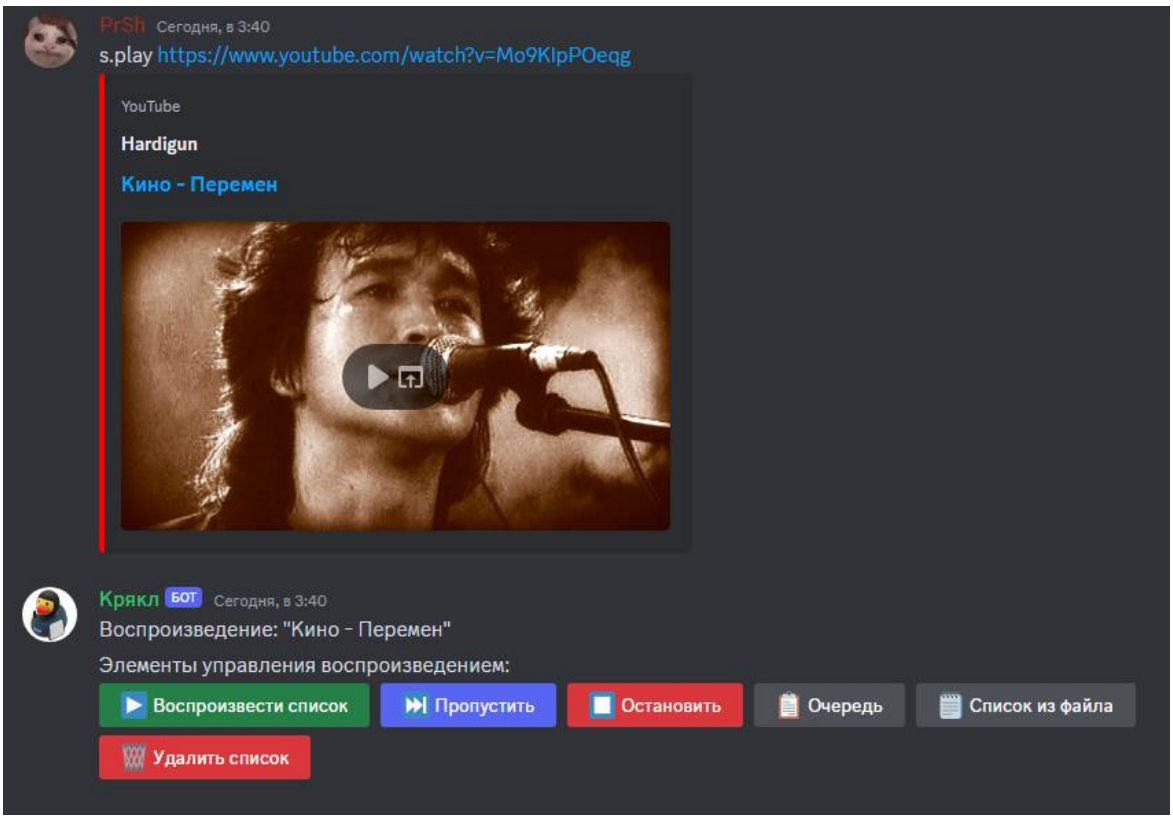


Рисунок 22 – Результат работы функции play по URL ссылке

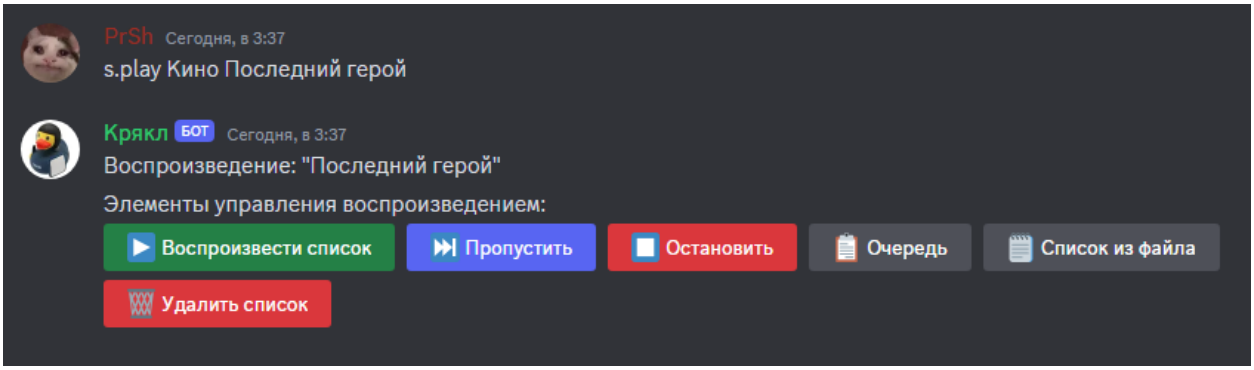


Рисунок 23 – Результат работы функции play по текстовому запросу

Самым главным обновлением музыкальной составляющей, по сравнению с прошлой курсовой работой, стало добавление возможности взаимодействовать не только через команды, а через функциональные кнопки, что намного приятнее и удобнее для любого пользователя. Добавлены такие кнопки, как: воспроизвести список, пропустить, остановить, очередь, список из файла, удалить список:

– воспроизвести список:

Позволяет начать проигрывание треков с первого по списку в очереди.

– пропустить:

Позволяет пропустить трек, который в данный момент проигрывается на следующий по списку.

– остановить:

Позволяет остановить трек, который в данный момент проигрывается.

– очередь:

Выводит всю существующую, на данный момент, очередь из добавленных ранее треков.

– список из файла:

Показывает список треков из локального файла.

– удалить список:

Удаляет все треки из очереди ранее добавленных треков.

Ну и конечно добавлена возможность взаимодействия с сервисом искусственного интеллекта OpenAI и сторонним сервисом, который выдает случайное фото собаки, для получения которой пользователю всего лишь нужно вписать команду s.rdog. А уже для взаимодействия с сервисом искусственного интеллекта OpenAI предоставлено две команды: s.news – выдает 5 последних найденных новостей по заданному запросу от пользователя и s.weather – отображает погоду в заданном городе. Реализация отображена на рисунке 24, а результат работы на рисунках 25, 26, 27:

```python
@bot.command(help="Эта команда отправляет случайное изображение собачки. Используйте: s.rdog")
async def rdog(ctx):
    try:
        const = requests.get("https://random.dog/woof.json")
        stuff = json.loads(const.text)
        embed = discord.Embed(title=f"URL: {stuff['url']}", color = discord.Color.random())
        embed.set_image(url=f"{stuff['url']}")
        await ctx.send(embed=embed)
    except Exception as e:
        logging.error(f"Ошибка в {ctx.command}: {e}")
        await ctx.send(f"Пожалуйста, убедитесь, что у вас есть подключение к Интернету")


@bot.command(help="Эта команда отображает последние новости по заданному ключевому слову. Используйте: s.news [Запрос]")
async def news(ctx, innews):
    news = requests.get(f"https://newsapi.org/v2/everything?q={innews}&apiKey={news_key}").json()
    thnews = news["articles"]
    if news["status"] == "ok":
        for i , article in enumerate(thnews):
            if i < 5:
                emend = discord.Embed(title=article["title"], description=article["description"], url=article["url"],color=discord.Color.blue())
                await ctx.send(embed=emend)
            else:
                break
    else:
        await ctx.send(f"{innews} не найдено, пожалуйста, попробуйте еще раз")


@bot.command(help="Эта команда отображает погоду в заданном городе. Используйте: s.weather [Название города]")
async def weather(ctx, *, city: str = None):
    try:
        weather_data = requests.get(f"https://api.openweathermap.org/data/2.5/weather?q={city}&units=imperial&APPID={weather_key}")
        if weather_data.json()['cod'] == '404':
            await ctx.send(f"Город {city} не найден")
        weather = weather_data.json()['weather'][0]['main']
        temp = round(weather_data.json()['main']['temp'])
        country_name = weather_data.json()['sys']['country']
        celsius = (temp - 32) * 5/9
        int_celsius = int(celsius)
        await ctx.send(f"""
    Погода в городе {city}  {weather} , страна: {country_name}
    Температура в городе {city} : {int_celsius}°C
    """)
    except Exception as e:
        logging.error(f"Ошибка в {ctx.command}: {e}")
```

Рисунок 24 – Код команд rdog, news и weather

Рисунок 25 – Результат команды rdog

Рисунок 26 – Результат команды news

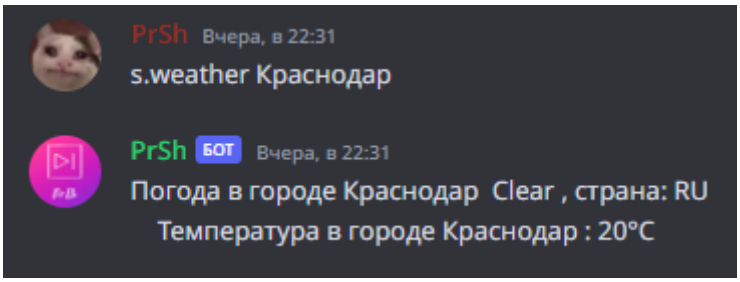


Рисунок 27 – Результат команды weather

Также добавлено взаимодействие с популярным сервисом искусственного интеллекта Hugging Face Api – американская компания, которая разрабатывает инструменты для создания приложений с использованием машинного обучения, имеющий огромную базу данных для генерации изображений, моделей, наборов данных, таблиц, пространств, сообщений, документов, решений, ценообразований. В моем Discord боте реализована генерация изображений по заданному запросу от пользователя посредством этих инструментов и баз данных, с помощью команды art. Реализация видна на рисунке 28, а результат работы на рисунке 29:

```python
@bot.command(name='art')
async def generate_image(ctx, *, prompt):
    api_url = "https://api-inference.huggingface.co/models/stabilityai/stable-diffusion-xl-base-1.0"
    headers = {"Authorization": f"Bearer {HUGGING_FACE_API_TOKEN}"}

    #payload = {
    #    "inputs": prompt,
    #    "options": {
    #        "wait_for_model": True
    #    }
    #}
    def query(payload):
        response = requests.post(api_url, headers=headers, json=payload)

        return response.content

    try:

        bytes = query(
            {
                "inputs": prompt
            }
        )
        import io
        from PIL import Image
        image = Image.open(io.BytesIO(bytes))
        with io.BytesIO() as image_binary:
            image.save(image_binary, 'JPEG')
            image_binary.seek(0)
            await ctx.send(file=discord.File(fp=image_binary, filename='generated_image.jpg'))


    except requests.exceptions.RequestException as e:
        print(f"Ошибка запроса API: {e}")
        await ctx.send("Произошла ошибка при выполнении запроса к API.")

    except json.JSONDecodeError as e:
        print(f"Ошибка декодирования JSON: {e}")
        await ctx.send("Произошла ошибка при выполнении запроса к API.")
```

Рисунок 28 – Реализация команды art

Рисунок 29 – Результат работы команды art

Реализовано подключение Wikipedia Api, которое с помощью искусственного интеллекта находит по заданному запросу информацию с всемирной онлайн энциклопедии Wikipedia, изучает и обрабатывает всю страницу, после чего тезисно или кратко выдает пользователю в чат для удобного и быстрого ознакомления. Реализация видна на рисунке 30, а вывод на рисунках 31 и 32:

```python
@bot.command()
async def askwiki(ctx, *, query):
    try:
        headers = {'User-Agent': 'StarloExo Bot/1.0 (Discord Bot)'}
        wiki_wiki = wikipediaapi.Wikipedia('ru', language='ru', headers=headers)
        page = wiki_wiki.page(query)
        page_summary = page.summary

        if page_summary:
            image_url = f"https://ru.wikipedia.org/wiki/File:{page.title.replace(' ', '_')}.png"

            embed = discord.Embed(title=query, description=page_summary)
            embed.set_image(url=image_url)
            await ctx.send(embed=embed)
        else:
            await ctx.send("No Wikipedia page found for the given query.")
    except json.JSONDecodeError:
        await ctx.send("Error: Invalid JSON response from the Wikipedia API.")
    except Exception as e:
        await ctx.send(f"Error: {str(e)}")
```

Рисунок 30 – Реализация Wikipedia Api и команды askwiki

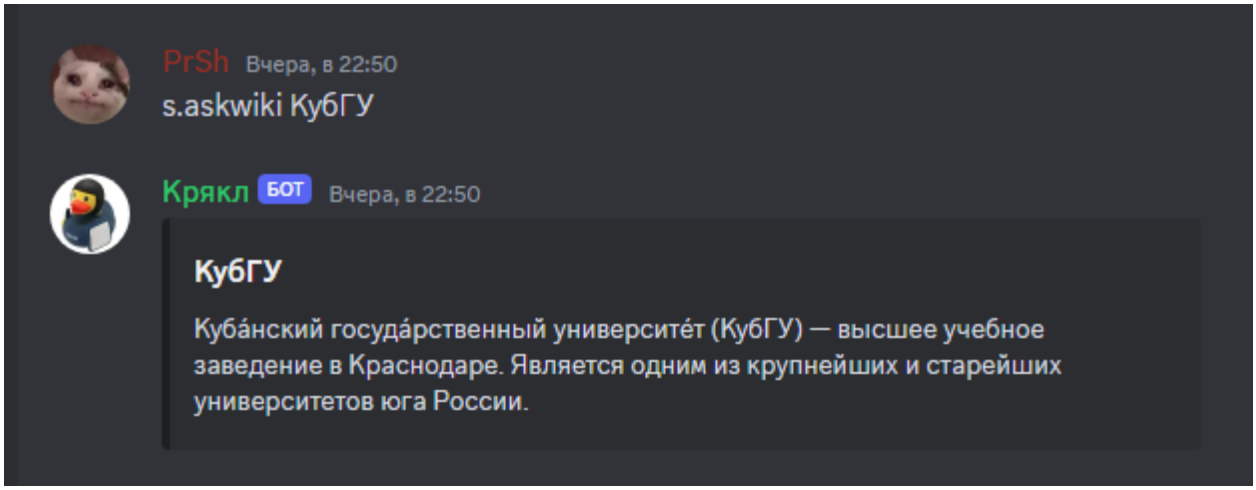


Рисунок 31 – Результат работы Wikipedia Api и команды askwiki

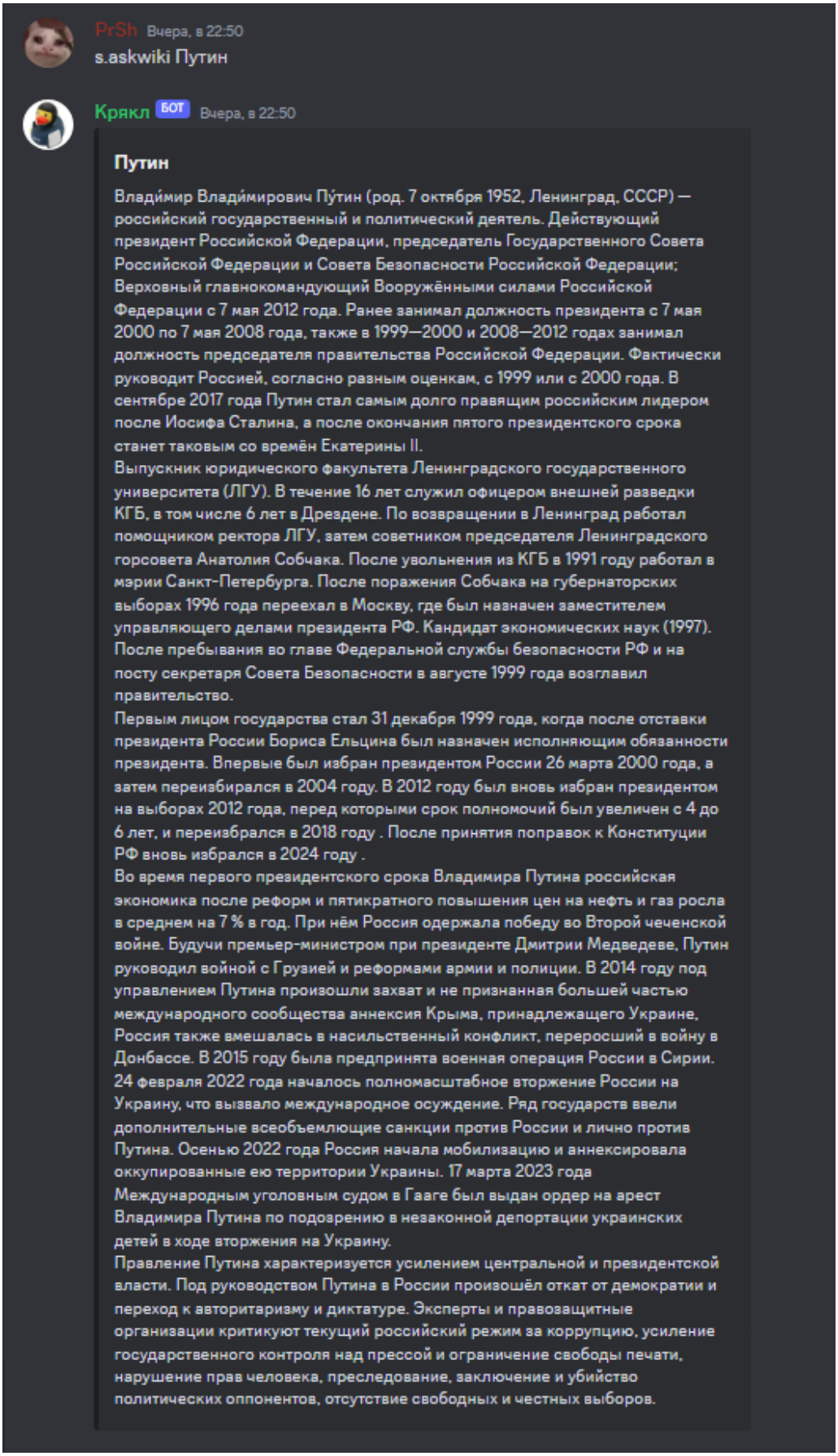


Рисунок 32 – Результат работы Wikipedia Api и команды askwiki

Осуществлен и вывод информации о сервере, который позволяет пользователям быстро узнать краткую информацию такую, как: логотип сервера, название сервера, ID сервера, владелец сервера, количество участников на сервере и дату создания. Реализация на рисунке 33, вывод на рисунке 34:

```python
@bot.command()
async def serverstats(ctx):
    guild = ctx.guild

    embed = discord.Embed(title="Статистика Сервера:", color=discord.Color.green())

    embed.set_thumbnail(url=guild.icon.url)

    embed.add_field(name="Имя Сервера:", value=guild.name, inline=True)
    embed.add_field(name="ID Сервера:", value=guild.id, inline=True)
    embed.add_field(name="Владелец:", value=guild.owner.name if guild.owner else "Unknown", inline=True)
    embed.add_field(name="Кол-во Участников:", value=guild.member_count, inline=True)
    embed.add_field(name="Дата Создания:", value=guild.created_at.strftime("%d-%m-%Y %H:%M:%S"), inline=True)

    await ctx.send(embed=embed)

```

Рисунок 33 – Реализация вывода информации о сервере

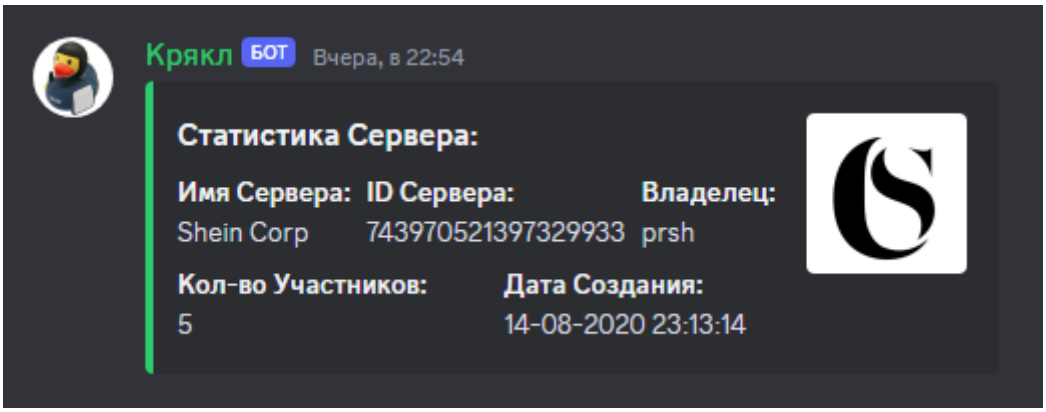


Рисунок 34 – Вывод информации о сервере

Добавлены команды, которые работают исходя из внутренних возможностей языка программирования Python, таких как randint() и eval(expression). Команда rnum отправляет случайное число в диапазоне от 1 до заданного числа, а команда math – это простой калькулятор. Реализация отображена на рисунке 35, а результат работы на рисунках 36, 37:

```python
@bot.command(help="Эта команда отправляет случайное число в диапазоне от 1 до нужного Вам случайного числа. Используйте: s.rnum [Число]")
async def rnum(ctx , *, randeom_num: int = None):
    try:
        randrom_number = random.randint(1, randeom_num)
        await ctx.send(f"Ваше случайное число {randrom_number} ~ {ctx.message.author.mention}")
    except ValueError:
        await ctx.send(f"{randeom_num} это не номер, пожалуйста, повторите попытку с действительным номером {ctx.message.author.mention}")


@bot.command(help="Простой калькулятор. Используйте: s.math [Число + | - | * | /  Число]")
async def math(ctx, *,  expression: str = None):
    try:
        await ctx.send(f"{eval(expression)}")
    except SyntaxError:
        await ctx.send(f"{expression} не является допустимым выражением, поддерживаются выражения + для сложения, - для вычитания, * для умножения и / для деления ~ {ctx.message.author.mention}")
    except NameError:
        await ctx.send(f"{expression} Ошибка в имени, пожалуйста убедитесь, что вы используете правильный синтаксис ~ {ctx.message.author.mention}")
    except ZeroDivisionError:
        await ctx.send(f"Делить на 0 нельзя! {ctx.message.author.mention}")
```

Рисунок 35 – Код команд rnum и math

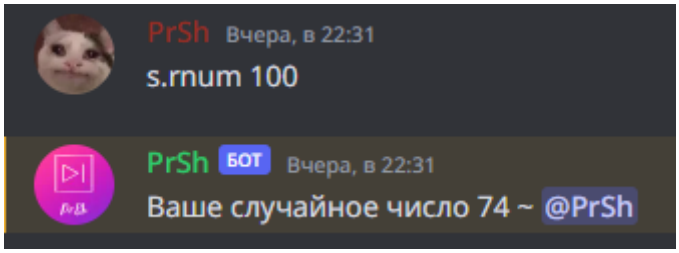


Рисунок 36 – Результат команды rnum

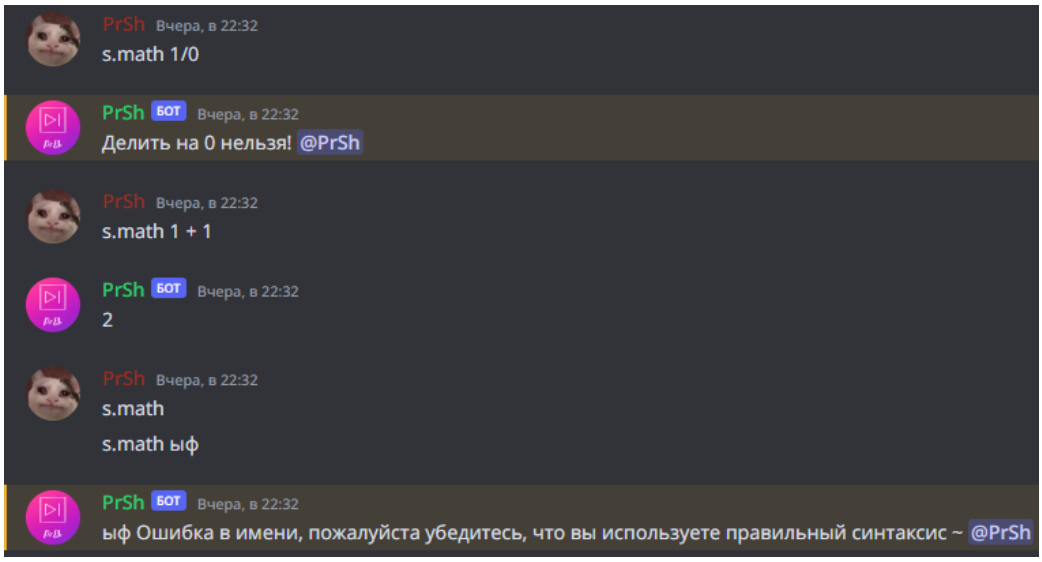


Рисунок 37 – Результат команды math

ЗАКЛЮЧЕНИЕ

Системы искусственного интеллекта, работающие на основе нейронных сетей, показывают большие успехи в анализе и классификации огромных наборов данных. Они позволяют автоматизировать различные процессы деятельности человека. В результате, находятся новые области для применения таких систем и методы оптимизации алгоритмов, разработанных для решения различных прикладных задач.

В данной выпускной квалификационной работе рассмотрены основные принципы работы искусственных нейронных сетей: строение, виды, методы обучения, сбора и анализа данных. На основе изученной информации создана теоретическая модель нейронной сети для работы бота, который решает задачи модерации текстовых каналов, получения информации о пользователе или сервере, настройка приветствия и прощания с участниками, управление музыкой.

В данной выпускной квалификационной работе осуществлена реализация Discord бота на языке программирования Python, произведена работа по подключению нейронных сетей и внутреннего функционала языка программирования Python в доступный список команд, произведена оценка возможности будущего продвижения проекта на рынок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Обучение нейросети с учителем, без учителя, с подкреплением – в чем отличие? Какой алгоритм лучше: сайт. – 2024. – URL:https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/ (дата обращения: 01.05.2024).

2. От чат-ботов к разговорному ИИ: разработка умных ассистентов для бизнеса: сайт. – 2024. – URL:https://vc.ru/promo/42838-aimylogic (дата обращения: 10.05.2024).

3. От чат-ботов к разговорному искусственному интеллекту: сайт. – 2024. – URL:https://konveier.com/blog/2018/08/02/ot-chat-botov-k-razgovornomu-iskusstvennomu-intellektu/ (дата обращения: 01.05.2024).

4. Что такое нейронная сеть: сайт. – 2024. – URL:https://ru.wikipedia.org/wiki/Нейронная_сеть (дата обращения: 01.05.2024).

5. Зенин, А. В. Исследование возможностей использования нейронных сетей / А.В.Зенин // Журнал молодой ученый. – 2017. – №16(150). – С. 130–140.

6. Рашид, Т. Создаем нейронную сеть / Т.Рашид. – Пер. с англ. – Спб.: ООО «Альфа-книга», 2017. – 272 с.

7. Николенко, С. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. – СПб.: Питер, 2019. – 480 с.

8. Haykin, S. Neural networks and learning machines / S. Haykin. – Hamilton: Neural Networks and Learning Machines Third Edition – 2009. – 938 p.

9. Элбон, К. Машинное обучение с использованием Python. Сборник рецептов / К. Элбон. – Пер. с англ. – СПБ.: БХВ-Петербург, 2019. – 384 с.

10. Горбань, А. Н. Обобщённая аппроксимационная теорема и вычислительные возможности нейронных сетей. Архивная копия от 27 января

2012 на Wayback Machine / А. Н. Горбань // Сибирский журнал вычислительной математики. – 1998. – Т. 1, № 1. – С. 12–24.

11. Кенин, А. М., Мазуров В. Д., Д. Р. Первушин Опыт применения нейронных сетей в экономических задачах: сайт. – 2024. – URL: https://web.archive.org/web/20130402025015/http://www.uralstars.com/Docs/Editor/Neuro.htm (дата обращения: 10.05.2024).

12. Руководство по использованию библиотеки discord.py: сайт. – 2024. – URL:https://github.com/denisnumb/discord-py-guide/blob/main/discord-py.md (дата обращения: 10.05.2024).

13. Создаем Discord-бота на Python: сайт. – 2024. – URL:https://tproger.ru/articles/sozdajom-discord-bota-na-python/ (дата обращения: 10.05.2024).

14. Discord: как создавался популярный мессенджер и в чем его успех: сайт. – 2024. – URL:https://streampark.ru/blog/discord-kak-sozdavalsya-populyarnyj-messendzher-i-v-chem-ego-uspeh/ (дата обращения: 02.06.2024).

2024-06-03 02:01:01

Уникальность текста: 81%

Источник	Сходство %
https://vc.ru/42838-aimylogic?from=rss	7%
https://ai-news.ru/2018/07/ot_chat_botov_k_razgovornomu_ii_razrabotka_umnyh_assistentov_dlya_biznesa.html	6%
https://genby.livejournal.com/684276.html?page=1	3%
https://genby.livejournal.com/684276.html	3%
https://www.trancearea.com/chto-takoe-gipoteticheski/	2%
http://chipgu.ru/viewtopic.php?t=3002&start=70	2%
https://ed.kyrg.info/daydzhest-vuzov-partnerov-60/	2%
https://rscf.ru/prjcard_int?22-15-00011	2%
https://xn-----6kcjd7aa0cfnmaec4e.xn--p1ai/решено-согласны-ли-вы-с-суждением-и-ф-а/	2%
https://pikabu.ru/story/a_vyi_znali_chto_odin_iz_glavnyikh_geroev_6ogo_sezona_dekstera_syin_toma_khyenksa_129…	2%
https://genby.livejournal.com/684276.html?page=2	2%
https://studfile.net/preview/4658895/page:31/	2%
https://www.rusnor.org/pubs/articles/10763.htm	2%
https://www.audit-it.ru/club/user/62449/blog/10192/	2%
https://elesta.ru/arm-do-yupiter-istoriya-izmeneniy	1%
https://www.drupal.org/project/ulogin/issues/1795036	1%
https://www.windowsnoticias.com/ru/изменить-имя-пользователя-виндовс-11/	1%
https://vk.com/wall312197125_29	1%
https://ru.stackoverflow.com/questions/860936/Как-создать-массив-такого-вида	1%
http://cuntroll.ru/article18/	1%
https://cccp3d.ru/topic/69391-заполнение-свойств-детали/	1%