1 ВВЕДЕНИЕ И ОБЗОР

Эта книга призвана помочь инженерам машинного обучения (ML) и специалистам по data science успешно пройти собеседование по проектированию ML-систем с акцентом на генеративный ИИ (GenAI). Она дополняет предыдущую книгу «ML System Design Interview»[1] [1], в которой рассматриваются стандартные, но фундаментальные темы, такие как поисковые и рекомендательные системы. Эта книга посвящена приложениям GenAI и уникальным проблемам проектирования таких систем. Она также призвана служить руководством для тех, кто хочет понять, как GenAI применяется на практике.

В этой главе рассматриваются две ключевые темы. Первая — это обзор GenAI с погружением в его фундаментальные концепции и области применения. Вторая — введение в полноценный фреймворк для построения ML-систем, который необходим для реальных сценариев использования и для подготовки к собеседованию. Этот фреймворк послужит основой для разработки популярных систем GenAI в следующих главах.

Итак, приступим.

Обзор GenAI

Искусственный интеллект (ИИ) — область computer science, занимающаяся созданием систем, способных выполнять задачи, обычно требующие человеческого интеллекта, такие как рассуждение, планирование и решение проблем. ML — это подраздел ИИ, использующий алгоритмы для обучения на основе данных, а не предопределенные правила. Эти алгоритмы анализируют данные, выявляют закономерности и делают прогнозы или генерируют новый контент на основе изученных закономерностей. Такие приложения, как рекомендательные системы, системы обнаружения мошеннических действий, автономные транспортные средства и чат-боты, обычно работают на основе ML-моделей.

ML-модели, как правило, делятся на две категории:

- дискриминативные;
- генеративные.

[1] Сюй А., Аминиан А. «System Design. Машинное обучение. Подготовка к сложному интервью». СПб., издательство «Питер».

Рис. 1.1. Взаимосвязь между ИИ и ML

Дискриминативные модели

Дискриминативные модели классифицируют данные, анализируя различия между классами на основе входных признаков. Формально они изучают условные вероятности, $P(Y|X)$, где Y — целевая переменная, а X — входные признаки.

Дискриминативные модели могут использоваться как для классификации, когда надо определить класс, к которому принадлежат входные данные, так и для регрессии, когда цель состоит в предсказании непрерывного значения. Например, при обнаружении мошеннических действий дискриминативная модель может классифицировать транзакции как легитимные или мошеннические на основе анализа таких характеристик, как сумма транзакции и история покупок. Другой пример: рекомендательная модель фильмов предсказывает, какой рейтинг присвоит пользователь картине, на основе истории его взаимодействий.

Распространенные алгоритмы разработки дискриминативных моделей включают следующие:

- **логистическая регрессия** (logistic regression): линейная модель, которая предсказывает вероятность бинарного исхода на основе входных признаков;
- **метод опорных векторов** (support vector machine, SVM): SVM находит гиперплоскости, которые наилучшим образом разделяют классы в пространстве признаков. Алгоритм может быть расширен для случая нелинейных границ с помощью функций ядра [2];
- **деревья решений** (decision trees): эти модели и их разновидности, такие как случайный лес, рекурсивно разбивают данные на подгруппы на основе целевой переменной;
- **метод k-ближайших соседей** (K-nearest neighbors, KNN): непараметрический метод, который классифицирует образец на основе большинства меток среди его ближайших соседей в пространстве признаков;

- **нейронные сети** (neural networks): эти модели состоят из слоев взаимосвязанных нейронов. Они используют взвешенные входные данные, функции активации и обратное распространение для обучения и аппроксимации сложных функций в таких задачах, как классификация и регрессия.

Хотя эти алгоритмы могут предсказывать целевую переменную по входным признакам, большинству из них не хватает способности изучать исходное распределение данных для генерации новых экземпляров данных. Для этого мы обращаемся к генеративным моделям.

Генеративные модели

Генеративные модели нацелены на понимание и воспроизведение исходного распределения входных данных. Формально они моделируют распределение $P(X)$, когда во внимание принимаются только входные данные (например, генерация изображений), или совместное распределение вероятностей $P(X, Y)$, когда рассматриваются и входные данные, и целевая переменная (например, преобразование текста в изображение). Это позволяет генерировать новые экземпляры данных путем выборки из изученных распределений.

В отличие от дискриминативных моделей, которые нацелены на различение экземпляров данных, генеративные могут создавать новые образцы (семплы) данных, которые очень похожи на исходные. Например, генеративная модель, обученная на изображениях человеческих лиц, может генерировать совершенно новые лица. Эти модели применяются в различных задачах, например генерации текстов, изображений и синтезе речи.

Генеративные алгоритмы можно разделить на две категории: классические и современные. Классические алгоритмы хорошо справляются с изучением закономерностей на основе структурированных данных. Однако им сложно обучаться на более сложных или неструктурированных данных. К распространенным классическим генеративным алгоритмам относятся:

- **наивный байесовский классификатор** (naïve Bayes): вероятностная модель, основанная на теореме Байеса [3];
- **модель гауссовых смесей** (Gaussian mixture model, GMM): GMM [4] представляет данные как смесь гауссовых распределений;
- **скрытая марковская модель** (hidden Markov model, HMM): HMM [5] моделирует совместные вероятности наблюдаемых последовательностей и скрытых состояний, порождающих эти последовательности;
- **машина Больцмана** (Boltzmann machine): модель на основе энергии, используемая для изучения признаков или уменьшения размерности [6].

С другой стороны, современные генеративные алгоритмы обучаются на основе сложных распределений данных и хорошо подходят для таких задач, как создание реалистичных изображений и генерация точных текстовых результатов в ответ на запросы. К распространенным современным генеративным алгоритмам относятся:

- **вариационные автоэнкодеры** (variational autoencoder, VAE): тип автоэнкодера, который моделирует распределение данных, кодируя их в скрытое пространство и затем восстанавливая исходные данные с помощью декодера;
- **генеративно-состязательные сети** (generative adversarial network, GAN): класс нейронных сетей, в которых генератор и дискриминатор обучаются одновременно. Генератор создает реалистичные данные, а дискриминатор пытается отличить реальные данные от сгенерированных;
- **диффузионные модели** (diffusion model): модели, обучающиеся сложным распределениям данных с помощью обратной диффузии. Они широко используются для создания изображений и видео;
- **модели авторегрессии** (autoregressive model): модели, которые генерируют данные, предсказывая каждый элемент последовательности на основе предыдущих элементов. Они широко используются для создания текстов и прогнозирования временны́х рядов.

Дискриминативные и генеративные модели применяются для разных целей. Дискриминативные модели обычно используются для классификации или прогнозирования, а генеративные — для генерации новых образцов. На рис. 1.2 показаны популярные задачи, в которых применяются генеративные и дискриминативные модели.

Рис. 1.2. Популярные задачи, решаемые с использованием ML

Что такое GenAI и почему он набирает популярность?

GenAI предполагает использование современных генеративных алгоритмов для обучения моделей, способных создавать новые экземпляры данных, например изображения, видео, текст и аудио.

GenAI приобрел большую популярность по двум основным причинам. Во-первых, эти модели могут выполнять всевозможные задачи в разных областях: генерировать текст, создавать реалистичные изображения и сочинять музыку. Такая многозадачность делает их ценными во всех сферах, от творчества и развлечений до здравоохранения и разработки программного обеспечения.

Во-вторых, приложения GenAI значительно повышают производительность. Например, при создании контента эти модели могут генерировать черновики текстов, предлагать улучшения или даже создавать конечные продукты, экономя значительное время и ресурсы. Другой пример — использование больших языковых моделей (large language model, LLM), таких как ChatGPT [7], которые могут отвечать на сложные вопросы и вести содержательные диалоги. В недавнем отчете McKinsey [8] прогнозируется, что GenAI обеспечит рост производительности труда на 0,1–0,6 % в год до 2040 года.

Почему GenAI становится таким мощным?

Модели GenAI демонстрируют впечатляющие возможности и очень мощно развиваются. Этому способствуют три ключевых фактора:

1) данные;
2) объем модели;
3) вычисления.

Данные

Эффективность ML-модели зависит от исходных данных для обучения. Например, если модель не обучена на обширных медицинских данных, она не справится с точной диагностикой заболеваний. Улучшение модели для конкретной задачи требует больших наборов данных с метками, но сбор таких данных может быть сложным и дорогостоящим.

Одним из ключевых условий успеха GenAI является самообучение. В отличие от классических моделей, которые обычно хорошо работают при обучении на размеченных данных, модели GenAI могут обучаться на неразмеченных данных. Такой подход позволяет им использовать обширные наборы данных из интернета без дорогостоящих и трудоемких процессов разметки.

Благодаря легкому доступу к очень большим массивам данных из интернета современные модели GenAI можно обучать на огромных датасетах, иногда превы-

шающих миллиарды текстовых документов или изображений. Например, модель Llama 3 компании Meta [9] была обучена на 15 триллионах лексем — примерно 50 терабайтах данных; модель Flamingo компании Google [10] была обучена на 1,8 миллиарда пар «изображение — текст». Обучение на таком огромном объеме данных помогает этим моделям изучать сложные закономерности и нюансы, что дает высококачественные результаты.

Рис. 1.3. Сравнение объема данных для обучения чат-бота и для диагностики заболеваний

Объем модели

Еще одним ключевым условием эффективности ML-моделей является их способность к обучению. Она измеряется двумя способами:

- количеством параметров;
- количеством FLOP.

Количество параметров

Параметры — это значения в модели, полученные в процессе обучения. Количество параметров является ключевым показателем способности модели обучаться на данных.

Модель с большим количеством параметров, как правило, обладает лучшей способностью к изучению сложных закономерностей и взаимосвязей, существующих в данных. Это часто ведет к повышению производительности, если модель была обучена на большом датасете. В табл. 1.1 приведены пять популярных моделей и количество их параметров.

Таблица 1.1. Популярные модели GenAI и количество их параметров[1]

Название модели	Параметры
PaLM от Google [11]	540B
GPT-3 от OpenAI [12]	175B
Flamingo от Google [10]	80B
Llama 3 от Meta [9]	405B
Imagen от Google [13]	2B

Количество FLOP

Метрика FLOP (Floating Point Operations) измеряет вычислительную сложность модели, подсчитывая количество операций с плавающей точкой, необходимых для выполнения одного прямого прохода. Сюда входят основные арифметические операции, такие как сложение, умножение и другие, которые выполняются по мере перемещения данных по слоям модели.

Чтобы лучше понять метрику FLOP, рассмотрим простой пример.

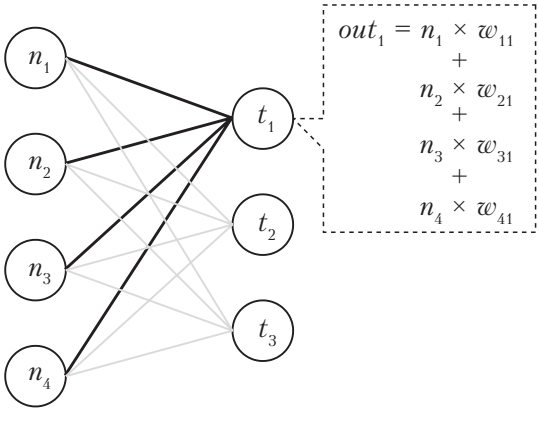

Рис. 1.4. Простой полносвязный слой и арифметические вычисления для одного выходного нейрона

Рассмотрим полносвязный слой с четырьмя входными и тремя выходными нейронами. Каждый выходной нейрон вычисляется путем умножения входных нейронов на соответствующий им вес и их суммирования. Таким образом, полу-

[1] B — от billion (миллиард). «Модель 540B» — модель с количеством параметров, равным 540 млрд. — *Примеч. ред.*

чается четыре операции умножения и три операции сложения для каждого выходного нейрона, как показано на рис. 1.4. А общее количество FLOP составляет $3 \times (4 + 3) = 21$.

В то время как количество параметров измеряет размер модели, FLOP указывает на количество арифметических вычислений и дает представление о вычислительной сложности модели. Хотя у модели с большим количеством параметров количество FLOP обычно больше, это не всегда так. Решающую роль играет архитектура. Например, плотные слои обычно требуют больше FLOP, чем разреженные соединения, даже если количество параметров одинаково. Понимание этого различия очень важно при оптимизации модели, поскольку оно помогает разрабатывать модели, которые одновременно точны и эффективны с вычислительной точки зрения.

Вычисления

С увеличением емкости моделей их производительность, как правило, повышается, но обучение таких больших моделей требует огромных вычислительных ресурсов. Вычисления, необходимые для обучения модели, часто измеряются во FLOP, то есть в общем количестве выполняемых операций. Например, для обучения модели PaLM-2 компании Google было использовано 10^{22} FLOP [14].

Вычислительная мощность обычно обеспечивается такими аппаратными средствами, как центральные процессоры (CPU), графические процессоры (GPU) и тензорные процессоры (TPU). Nvidia, например, предлагает такие передовые GPU, как H100, A100 и A10, каждый из которых отличается стоимостью и вычислительными возможностями. Производительность этих машин часто измеряется во FLOP/S (операции с плавающей точкой в секунду). Например, GPU Nvidia H100 может производить до 60 терафлопс в секунду (60 TFLOP/S) [15].

Обучение продвинутых моделей GenAI стоит дорого и требует использования тысяч графических процессоров в течение нескольких недель. Чтобы понять требования к вычислениям для таких моделей, как PaLM-2, рассчитаем необходимое количество графических процессоров H100. Если предположить, что пиковая производительность H100 составляет 60 TFLOP/S, то он может выполнять примерно $5{,}18 \times 10^{18}$ FLOP в день. Учитывая, что для PaLM-2 требуется 10^{22} FLOP, одному графическому процессору H100 потребуется около 5,5 года для выполнения необходимых вычислений. Поэтому стоимость обучения больших моделей чрезвычайно высока и часто превышает десятки миллионов долларов. Например, Сэм Альтман (Sam Altman), генеральный директор OpenAI, заявил, что стоимость обучения GPT-4 составила более 100 миллионов долларов [16].

Обучение модели с миллиардами параметров (например, GPT-4, Llama 3) было невозможно всего несколько лет назад. Сдвиг произошел в основном благодаря развитию аппаратного обеспечения, в частности специализированного оборудования, такого как GPU и TPU, предназначенного для задач глубокого обучения.

Распределенное обучение также сыграло важную роль, позволив распараллелить рабочую нагрузку между тысячами машин. Это значительно ускоряет процесс, делая возможным обучение очень больших моделей на огромных датасетах. Благодаря этим усовершенствованиям инфраструктуры и методам обучения стало возможным обучать модели GenAI в беспрецедентных масштабах.

Закон масштабирования

Какова оптимальная комбинация размера модели и обучающих данных (измеряемых количеством лексем, или токенов), при которой в условиях ограниченного бюджета на вычисления (измеряемого во FLOP) потери будут минимальными? Это фундаментальный вопрос, на который исследователи пытаются ответить с помощью законов масштабирования.

В 2020 году исследователи OpenAI провели обширные эксперименты по обучению LLM, изучая различные факторы, такие как размер модели (N), размер набора данных (D), вычислительные ресурсы (C), архитектуры моделей и длина контекста [17]. Полученные результаты позволили сделать два ключевых вывода. Во-первых, влияние масштабирования на производительность модели значительно более выражено, чем влияние архитектурных вариаций. Во-вторых, при увеличении размера модели, объема датасета или вычислительных ресурсов производительность соответствующим и предсказуемым образом повышается, следуя тенденции степенного закона.

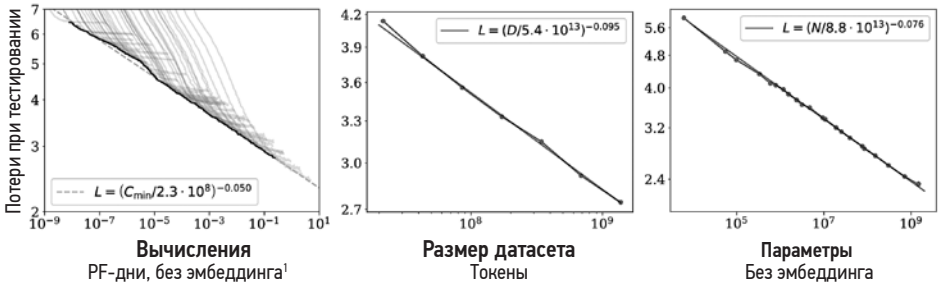

Рис. 1.5. Закон масштабирования OpenAI (см.: [17])

В 2022 году исследователи DeepMind пришли к выводу, что многие существующие LLM недообучены, то есть модели недостаточно велики для того объема данных, на котором они обучались [18]. Они обнаружили, что для достижения оптимальной производительности объем данных должен линейно увеличиваться с размером модели.

[1] PF-день = 10^{15} FLOP/с × 24 ч × 3600 с/ч, то есть 8,64 × 10^{19} FLOP. — *Примеч. ред.*

После недавнего выпуска модели o1 от OpenAI [19] исследователи начали выдвигать гипотезы о существовании закона масштабирования на этапе инференса [20].

Риски и ограничения GenAI

GenAI быстро развивается и способствует прогрессу во многих отраслях благодаря созданию реалистичных текстов, изображений и видео. Однако он также несет в себе серьезные риски и ограничения. Внимание к этим вопросам — важное условие ответственного и устойчивого развития. К таким проблемам относятся:

- **этические проблемы:** вопросы, связанные с ошибками смещения, интеллектуальной собственностью, дезинформацией и ненадлежащим использованием созданного контента, могут иметь пагубные последствия для общества;
- **воздействие на окружающую среду:** высокая вычислительная мощность, необходимая для обучения больших моделей, приводит к значительному потреблению энергии и выбросам углекислого газа;
- **ограничения модели:** моделям GenAI может не хватать истинного понимания, что приводит к неточностям и ограничениям в сложных задачах с рассуждениями, а также к галлюцинациям;
- **риски безопасности:** существуют угрозы, связанные с применением GenAI для создания дипфейков, используемых для шантажа, политических манипуляций, автоматизированных фишинговых атак и эксплойтов, манипулирующих результатами моделей в критически важных областях, таких как здравоохранение и финансы.

Все это серьезные проблемы разработки приложений GenAI. Устранение подобных рисков требует междисциплинарного подхода не только с привлечением технологий, но также с учетом этики, права и общественной сознательности.

Основа собеседований, посвященных проектированию ML-систем (ML System Design interview)

Многие инженеры считают, что алгоритмы ML — например, авторегрессионные модели трансформеров (transformer) или диффузионные модели — это и есть вся система ML. Однако создание и развертывание систем GenAI включает в себя гораздо больше, чем просто обучение модели. Эти системы сложны и включают в себя такие компоненты, как пайплайны данных для предварительной обработки больших датасетов и дальнейшей работы с ними, механизмы оценки качества и безопасности результатов, инфраструктура для доставки контента, созданного ИИ, в больших объемах и мониторинг для обеспечения стабильной работы с течением времени.

На собеседовании по разработке ML-систем, особенно ориентированных на GenAI, вы часто будете сталкиваться с открытыми вопросами. Например, вас могут попросить разработать чат-бота для работы с клиентами или инструмент для редактирования изображений с помощью ИИ. Универсального «правильного» ответа нет. Эксперта, проводящего интервью, интересует, как вы подходите к решению сложных задач, как вы понимаете концепции GenAI, видите процесс разработки системы и обосновываете свои решения.

Чтобы успешно пройти собеседование по разработке системы GenAI, очень важно придерживаться структуры. Бессвязные ответы могут затруднить понимание хода ваших мыслей и снизить ясность. В этой книге представлена схема, которая поможет пройти собеседование по проектированию системы GenAI. Схема включает следующие основные этапы.

1. Уточнение требований.
2. Представление проблемы в виде задачи ML.
3. Подготовка данных.
4. Разработка модели.
5. Оценка.
6. Общий дизайн ML-системы.
7. Развертывание и мониторинг.

Рис. 1.6. Этапы проектирования ML-системы

Рассмотрим каждый этап, чтобы проанализировать ключевые моменты и темы для обсуждения при разработке системы GenAI.

Уточнение требований

Когда вы начинаете разрабатывать ML-систему для решения конкретной задачи, у вас еще мало информации. Точно так же на собеседованиях вопросы о проектировании ML-систем часто звучат расплывчато и содержат минимум подробностей. Например, на собеседовании вас могут попросить «спроектировать систему генерации изображений». Первый шаг — задать уточняющие вопросы. Но какими они должны быть?

Вопросы должны прояснять проблему и цели, которые должна достичь система. Требования к системе делятся на два типа:

- функциональные;
- нефункциональные.

Функциональные требования

Функциональные требования описывают то, что должна делать система, — ее основные возможности. Например, «генерировать изображение и настраивать его стиль в зависимости от запроса пользователя» — это функциональное требование для системы преобразования текста в изображение. В контексте проектирования системы GenAI функциональные требования имеют решающее значение, поскольку они определяют высокоуровневую архитектуру системы. Они определяют направление разработки основных компонентов и функциональных возможностей, которые система должна предоставлять для удовлетворения потребностей пользователей.

Нефункциональные требования

Нефункциональные требования относятся к тому, *как* работает система, а не *что* она делает. В них входят такие метрики производительности, как время ожидания и пропускная способность, а также соображения справедливости, безопасности и масштабируемости. Например, в системе генерации изображений нефункциональные требования могут определять приемлемую скорость генерации изображений и стандарты качества, которым должны соответствовать полученные изображения. Несмотря на то что эти требования, как правило, относятся к категории продвинутых в проектировании систем GenAI и могут не сильно изменить исходную архитектуру, очень важно выявить и понять их на ранней стадии, поскольку от них часто зависят дальнейшие этапы проектирования, особенно при настройке производительности и улучшении системы.

Вот несколько вопросов для начала работы.

- **Бизнес-цель.** Какова основная цель системы? Какой конкретной цели она будет служить? Например, при разработке системы описания изображений необходимо знать, будет ли она использоваться для создания подробных описаний товаров на электронных торговых площадках или для предложения коротких подписей к фотографиям в соцсетях.
- **Функциональность системы.** Какие функции должна поддерживать система и как они могут повлиять на ML-дизайн? Например, при разработке системы генерации изображений важно знать, могут ли пользователи оставлять отзывы или оценивать сгенерированные изображения, так как эти взаимодействия могут улучшить модель. Аналогично, при разработке LLM важно знать, какие языки должны в ней поддерживаться.
- **Данные.** Каковы источники данных? Насколько велик набор данных? Размечены ли данные? Эти вопросы очень важны, поскольку качество и количество данных может повлиять на дизайн.
- **Ограничения.** Каковы доступные вычислительные ресурсы? Будет ли система облачной или сможет работать на локальном устройстве?

- **Масштаб системы.** Сколько пользователей, по оценкам, будут применять систему? Сколько изображений необходимо сгенерировать и каков ожидаемый рост спроса? Эти вопросы важно прояснить, поскольку система, предназначенная для создания изображений с расчетом на небольшую группу пользователей, не требует такого же уровня масштабируемости, как система, рассчитанная на обслуживание миллионов пользователей.
- **Производительность.** Как быстро должен генерироваться контент? Требуется ли генерация в режиме реального времени? Что важнее — качество контента или скорость его генерации?

Этот список не является исчерпывающим, но служит хорошей отправной точкой. Нельзя забывать и о других вопросах, таких как конфиденциальность, этика и безопасность данных.

К концу этого этапа интервью вы должны согласовать с экспертом область применения и требования к системе. Эти параметры важно уточнить, чтобы обеспечить соответствие ожиданиям эксперта.

Представление проблемы в виде задачи ML

Если эксперт просит вас разработать функцию для автоматического реферирования электронных писем, он ставит перед вами проблему. Но нельзя просто попросить ИИ реферировать электронные письма. Необходимо сформулировать проблему так, чтобы методы ИИ могли ее решить. Представление проблемы в виде задачи ML — ключевой шаг в проектировании ML-систем, так как на этом строится весь дизайн.

Для начала нужно определить, требуется ли вообще ML для решения проблемы. В системах GenAI ML, как правило, требуется, поскольку это основной инструмент разработки таких систем.

Следующие два шага помогут преобразовать проблему в задачу ML:

- определение входных и выходных данных системы;
- выбор подходящего метода ML.

Определение входных и выходных данных системы

Чтобы сформулировать проблему, сначала нужно определить входные и выходные данные системы. Для этого следует определить модальность входных данных (текст, изображение, аудио, видео) и ожидаемый выход. Например, в системе чат-бота входными данными является текстовый запрос пользователя, а выходными — ответ системы.

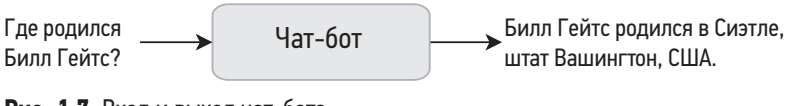

Рис. 1.7. Вход и выход чат-бота