ЛЕКЦИЯ 2
Инструменты и платформы искусственного интеллекта

Цель лекции

Цель лекции — сформировать у обучающихся представление о современных инструментах и программных платформах, используемых для разработки систем искусственного интеллекта, показать их назначение и место на различных этапах жизненного цикла AI-проекта, а также научить осознанно выбирать инструменты в зависимости от типа задачи, данных и требований к будущей интеллектуальной системе.

В соответствии с программой дисциплины практическая часть курса предусматривает работу с инструментами ИИ, создание моделей, их обучение, оценку и применение для решения прикладных задач. Поэтому важно понимать не только алгоритмы искусственного интеллекта, но и программную среду, в которой эти алгоритмы реализуются.

Что понимается под инструментами искусственного интеллекта

Разработка системы искусственного интеллекта представляет собой многоэтапный процесс. Прежде чем получить готовую модель, разработчик должен собрать данные, подготовить их, выбрать алгоритм, обучить модель, проверить качество её работы и затем интегрировать модель в программную систему.

Для выполнения этих задач используются различные программные средства.

Инструмент искусственного интеллекта — *это программное средство, библиотека, фреймворк, сервис или среда разработки, предназначенная для создания, обучения, тестирования, развертывания или использования моделей искусственного интеллекта.*

Например, разработчику необходимо создать модель, определяющую, является электронное письмо спамом или нет. Для этого ему потребуется загрузить набор сообщений, обработать текст, преобразовать его в числовое представление, обучить классификатор и проверить точность его работы.

Каждый из этих этапов можно выполнять с помощью специализированных программных инструментов.

Следовательно, термин «инструменты ИИ» охватывает значительно больше, чем отдельные приложения вроде чат-ботов. К этой категории относятся языки программирования, библиотеки машинного обучения, среды разработки, облачные сервисы, платформы для работы с моделями и средства визуализации данных.

Понятие платформы искусственного интеллекта

Наряду с отдельными инструментами используются комплексные платформы.

Платформа искусственного интеллекта — *программная или облачная среда, объединяющая набор средств для выполнения нескольких этапов разработки и использования интеллектуальных моделей.*

Основное отличие платформы от отдельного инструмента заключается в масштабе решаемых задач.

Например, отдельная библиотека может предоставлять алгоритмы машинного обучения. Платформа же может одновременно предоставлять вычислительную инфраструктуру, хранилище данных, средства обучения моделей, инструменты оценки их качества и механизмы развертывания.

Представим университетскую исследовательскую группу, которая разрабатывает модель классификации сетевого трафика. Исходные данные необходимо загрузить, очистить, исследовать и преобразовать. Затем нужно обучить несколько моделей, сравнить их и сохранить лучший вариант. После этого модель необходимо использовать для анализа новых данных.

Если все эти действия выполняются в единой программной среде, такая среда фактически становится платформой разработки AI-решения.

Жизненный цикл проекта искусственного интеллекта

Выбор инструментов становится понятнее, если рассмотреть типичный жизненный цикл проекта искусственного интеллекта.

Разработка AI-системы начинается не с обучения нейронной сети, а с постановки задачи.

Предположим, необходимо создать систему обнаружения фишинговых сообщений. Сначала требуется определить, что именно будет анализировать система и какой результат должна выдавать. Затем необходимо получить данные — например, набор обычных и фишинговых сообщений.

После получения данных начинается их подготовка. Необходимо проверить наличие пропущенных значений, удалить ошибочные или повторяющиеся записи, преобразовать текст и подготовить признаки для модели.

Следующий этап связан с выбором алгоритма и обучением модели.

После обучения необходимо оценить её качество. Недостаточно сказать, что модель «работает». Необходимо измерить результат с помощью соответствующих метрик.

После этого модель может быть интегрирована в приложение или сервис.

Таким образом, упрощённо процесс можно представить следующим образом:

Постановка задачи → Сбор данных → Подготовка данных → Обучение модели → Оценка → Развертывание → Мониторинг

На каждом этапе могут использоваться разные инструменты.

Python как основной язык разработки систем ИИ

Одним из наиболее распространённых языков программирования в области искусственного интеллекта является **Python**.

Python представляет собой универсальный язык программирования высокого уровня. Его популярность в искусственном интеллекте объясняется не только относительно простым синтаксисом, но прежде всего развитой экосистемой библиотек.

Разработчику не требуется самостоятельно программировать все математические операции или алгоритмы машинного обучения. Многие необходимые функции уже реализованы в специализированных библиотеках.

Например, если необходимо загрузить таблицу с данными о сетевых соединениях, удалить ненужные столбцы, обработать пропущенные значения и выполнить статистический анализ, для этого можно использовать готовые средства Python.

При этом сам Python не является системой искусственного интеллекта. Это язык программирования, на основе которого можно использовать различные AI-библиотеки и создавать собственные интеллектуальные приложения.

Для студентов Python особенно удобен тем, что один язык позволяет работать с данными, реализовывать классические алгоритмы машинного обучения, создавать нейронные сети, обрабатывать изображения и тексты.

Jupyter Notebook и интерактивная разработка

При изучении искусственного интеллекта часто используется **Jupyter Notebook**.

Jupyter Notebook — *интерактивная вычислительная среда, позволяющая объединять программный код, результаты его выполнения, текстовые пояснения, математические выражения и визуализации в одном документе.*

Это особенно удобно для анализа данных и машинного обучения.

В обычной программе разработчик пишет исходный код и запускает программу целиком. В Jupyter Notebook код можно разделить на небольшие блоки — ячейки — и выполнять их последовательно.

Например, в первой ячейке можно загрузить данные, во второй посмотреть их характеристики, в третьей выполнить очистку, в четвёртой построить график, а в пятой обучить модель.

Результат каждого действия сразу отображается под соответствующей ячейкой.

Поэтому Notebook удобен не только для программирования, но и для документирования эксперимента. Исследователь может видеть, какие операции выполнялись с данными и какие результаты были получены.

Google Colab как облачная среда

Для выполнения практических заданий по искусственному интеллекту удобно использовать облачные вычислительные среды.

Одним из распространённых вариантов является **Google Colab — облачная среда для выполнения Jupyter Notebook в браузере**.

Главное преимущество такого подхода состоит в том, что студенту не требуется полностью настраивать локальную среду разработки на собственном компьютере.

Код выполняется на удалённых вычислительных ресурсах.

Например, преподаватель может подготовить Notebook с практическим заданием. Студент открывает его, выполняет Python-код, изменяет параметры модели и анализирует результаты непосредственно через браузер.

Это особенно удобно при обучении нейронных сетей, поскольку для некоторых задач могут потребоваться более значительные вычислительные ресурсы.

При этом облачная среда имеет и ограничения. Пользователь зависит от подключения к сети, доступные вычислительные ресурсы могут быть ограничены, а при работе с конфиденциальными данными необходимо отдельно учитывать требования информационной безопасности.

NumPy и численные вычисления

Большинство алгоритмов искусственного интеллекта выполняют большое количество математических операций. Для эффективной работы с числовыми данными в Python широко используется библиотека **NumPy**.

NumPy — *библиотека Python, предназначенная для эффективной работы с многомерными числовыми массивами и выполнения математических операций над ними.*

Например, изображение для модели компьютерного зрения можно представить как числовой массив. Таблицу признаков объектов также можно преобразовать в матрицу чисел.

Если имеется набор данных о сетевом трафике, каждая строка может соответствовать одному соединению, а столбцы — длительности соединения, числу переданных пакетов, объёму данных и другим характеристикам.

Математические операции над такими массивами выполняются значительно удобнее с использованием специализированных библиотек, чем при ручной обработке каждого значения.

NumPy является фундаментом для многих других библиотек научных вычислений и машинного обучения в Python.

Pandas и работа с данными

Реальные данные часто представлены в табличной форме. Например, CSV-файл может содержать тысячи или миллионы строк наблюдений.

Для работы с такими данными используется библиотека **Pandas**.

Pandas — *библиотека Python, предназначенная для загрузки, обработки, преобразования и анализа структурированных данных.*

Одной из основных структур Pandas является **DataFrame** — табличная структура данных, состоящая из строк и столбцов.

Предположим, имеется набор данных о сетевых соединениях. В таблице содержатся IP-адрес, протокол, продолжительность соединения, число пакетов, объём переданных данных и метка класса.

Используя Pandas, можно загрузить этот набор данных, посмотреть первые записи, определить количество строк и столбцов, найти пропущенные значения, выбрать необходимые признаки и провести предварительный статистический анализ.

Таким образом, Pandas особенно важен на этапе подготовки данных.

Это необходимо подчеркнуть, поскольку качество модели искусственного интеллекта зависит не только от выбранного алгоритма. Если исходные данные содержат ошибки, пропуски или некорректные значения, даже сложная модель может показать неудовлетворительный результат.

Визуализация данных

Перед обучением модели данные необходимо исследовать. Одним из эффективных способов исследования является визуализация.

Визуализация данных — *графическое представление информации, позволяющее выявлять распределения, зависимости, выбросы и другие особенности данных.*

Например, необходимо определить, сбалансирован ли набор данных для классификации сетевых атак. Если в наборе имеется 95 000 примеров нормального трафика и только 5 000 примеров атак, возникает дисбаланс классов.

Графическое представление позволяет быстро увидеть эту проблему.

В Python для построения графиков широко применяются соответствующие библиотеки визуализации. На практике можно строить гистограммы, линейные графики, диаграммы рассеяния и другие виды представления данных.

Визуализация используется не только до обучения модели. С её помощью можно сравнивать результаты различных алгоритмов, анализировать ошибки и представлять результаты исследования.

Scikit-learn и классическое машинное обучение

Для изучения и практического применения классических методов машинного обучения широко используется библиотека **scikit-learn**.

Scikit-learn — *библиотека машинного обучения для Python, предоставляющая инструменты для предварительной обработки данных, обучения моделей, прогнозирования и оценки их качества.*

С её помощью можно решать задачи классификации, регрессии, кластеризации и снижения размерности.

Например, необходимо построить модель, которая по характеристикам сетевого соединения определяет, относится оно к нормальному трафику или атаке.

Сначала данные разделяются на обучающую и тестовую выборки.

Обучающая выборка используется для построения модели.

Тестовая выборка используется для проверки того, насколько хорошо обученная модель работает с данными, которые она не использовала непосредственно во время обучения.

Затем можно обучить несколько алгоритмов и сравнить результаты.

Например, использовать логистическую регрессию, дерево решений или метод опорных векторов.

Scikit-learn предоставляет единый подход к работе с различными алгоритмами, поэтому хорошо подходит для обучения основам машинного обучения. Согласно вашему силлабусу, последующие лекции посвящены обучению с учителем и без учителя,

классификации, регрессии, кластеризации и снижению размерности, поэтому данный инструмент непосредственно понадобится студентам в следующих темах.

TensorFlow и Keras

Для более сложных моделей, особенно нейронных сетей и глубокого обучения, используются специализированные фреймворки.

TensorFlow — *программная платформа для построения, обучения и применения моделей машинного и глубокого обучения.*

С помощью TensorFlow можно выполнять операции над многомерными тензорами, строить вычислительные модели и обучать нейронные сети.

С TensorFlow тесно связан **Keras**.

Keras — *высокоуровневый интерфейс для создания и обучения нейронных сетей, позволяющий описывать архитектуру модели с использованием относительно компактного программного кода.*

Например, при создании модели распознавания изображений разработчик может определить входной слой, несколько промежуточных слоёв и выходной слой, после чего задать параметры обучения.

Использование готового фреймворка освобождает разработчика от необходимости вручную реализовывать большое количество низкоуровневых математических операций.

В вашем силлабусе глубокое обучение, нейронные сети и архитектуры CNN, RNN, LSTM и Transformer рассматриваются в отдельной лекции, поэтому на данном этапе важно понять назначение соответствующих инструментов, а непосредственно архитектуры моделей изучать позже.

PyTorch

Другим широко используемым фреймворком глубокого обучения является **PyTorch**.

PyTorch предоставляет средства для выполнения операций над тензорами, автоматического вычисления градиентов, построения нейронных сетей и обучения моделей.

В исследовательской работе он удобен благодаря гибкому подходу к построению моделей и тесной интеграции с экосистемой Python.

Например, исследователь разрабатывает новую архитектуру нейронной сети для классификации сетевого трафика. Ему необходимо экспериментировать с количеством слоёв, изменять функции потерь, добавлять собственные компоненты и анализировать процесс обучения. Фреймворк глубокого обучения позволяет проводить такие эксперименты значительно быстрее, чем реализация всех операций с нуля.

Таким образом, TensorFlow/Keras и PyTorch решают во многом похожие задачи, однако выбор конкретного инструмента зависит от проекта, опыта команды, используемой инфраструктуры и требований к реализации.

Hugging Face и готовые модели искусственного интеллекта

Современная разработка ИИ не всегда требует обучения модели полностью с нуля.

Во многих случаях можно использовать предварительно обученные модели.

Предварительно обученная модель, pretrained model, — *модель, параметры которой уже были обучены на большом наборе данных и которая может использоваться непосредственно или адаптироваться для новой задачи.*

Это особенно распространено в обработке естественного языка и компьютерном зрении.

Одной из известных экосистем для работы с такими моделями является **Hugging Face**.

Она предоставляет инструменты и репозитории моделей, наборов данных и приложений машинного обучения.

Например, вместо создания языковой модели с нуля исследователь может выбрать предварительно обученную модель и адаптировать её для классификации текстов определённой предметной области.

Такой подход называется **transfer learning — перенос обучения**.

Его смысл заключается в использовании знаний, полученных моделью при решении одной большой задачи, для решения другой, более специализированной задачи.

Это позволяет уменьшить объём необходимых данных и вычислительных ресурсов.

Облачные платформы искусственного интеллекта

Для крупных проектов собственных вычислительных ресурсов может быть недостаточно. В этом случае используются облачные AI-платформы.

Облачная платформа искусственного интеллекта — *инфраструктура, предоставляющая через облачную среду вычислительные ресурсы и сервисы для подготовки данных, обучения, хранения, развертывания и использования моделей.*

Преимущество облачного подхода заключается в масштабируемости.

Например, небольшая модель может обучаться на обычном компьютере. Но если необходимо обработать очень большой набор изображений или обучить сложную нейронную сеть, могут потребоваться графические ускорители и значительный объём памяти.

Облачная инфраструктура позволяет временно использовать необходимые вычислительные ресурсы без покупки собственного оборудования.

Однако при использовании облачных платформ необходимо учитывать стоимость вычислений, зависимость от поставщика услуг и вопросы безопасности данных.

Генеративные AI-платформы

Отдельную категорию современных инструментов составляют платформы генеративного искусственного интеллекта.

Они предоставляют доступ к моделям, способным работать с текстом, программным кодом, изображениями и другими типами данных.

Пользователь может взаимодействовать с такой моделью через графический интерфейс либо программно через API.

API, Application Programming Interface, — *программный интерфейс, позволяющий одному приложению обращаться к функциям другого программного сервиса по установленным правилам.*

Например, разработчик создаёт интеллектуального помощника для университетской системы. Вместо разработки большой языковой модели с нуля он может использовать API готовой модели.

Пользователь задаёт вопрос через интерфейс приложения. Сервер формирует запрос к модели через API. Модель формирует ответ, после чего он возвращается пользователю.

Таким образом, готовая AI-модель становится одним из компонентов более крупной программной системы.

При этом использование внешней модели требует анализа вопросов конфиденциальности: какие данные передаются внешнему сервису, где они обрабатываются и допустима ли такая передача с точки зрения требований организации.

No-Code и Low-Code инструменты

Не все задачи искусственного интеллекта требуют написания большого объёма программного кода.

Существуют **No-Code** и **Low-Code** платформы.

No-Code — *подход, при котором приложение или модель создаётся преимущественно с помощью графических элементов без традиционного программирования.*

Low-Code — ***подход, при котором значительная часть разработки выполняется визуальными средствами, но при необходимости допускается использование программного кода.***

Например, пользователь может загрузить таблицу с данными, выбрать целевой столбец, указать тип задачи и запустить обучение нескольких моделей через графический интерфейс.

Такие инструменты удобны для быстрого прототипирования и предварительной проверки идеи.

Однако простота интерфейса не отменяет необходимости понимать методы машинного обучения.

Если пользователь не понимает, что такое обучающая и тестовая выборка, переобучение или дисбаланс классов, он может получить высокое значение некоторой метрики и ошибочно сделать вывод, что модель качественная.

Поэтому No-Code должен рассматриваться как средство автоматизации разработки, а не как замена знаниям в области искусственного интеллекта.

Выбор инструмента для конкретной задачи

Не существует одного универсального инструмента, который был бы лучшим для всех задач искусственного интеллекта.

Инструмент необходимо выбирать исходя из характера задачи.

Если студент только начинает изучать машинное обучение и работает с небольшим табличным набором данных, удобной комбинацией могут быть Python, Jupyter или Google Colab, Pandas и scikit-learn.

Если задача связана с глубокими нейронными сетями, могут использоваться PyTorch или TensorFlow.

При работе с готовыми языковыми моделями может быть полезна экосистема предварительно обученных моделей или API соответствующих AI-сервисов.

Если необходимо провести быстрый эксперимент без написания большого объёма программного кода, может использоваться Low-Code или No-Code платформа.

Следовательно, правильный вопрос заключается не в том:

«Какая AI-платформа самая лучшая?»

а в том:

«Какой инструмент наиболее подходит для конкретной задачи, данных, команды и условий эксплуатации?»

Критерии выбора AI-платформы

Выбор платформы необходимо выполнять осознанно.

Прежде всего учитывается тип решаемой задачи. Инструмент, удобный для обработки таблиц, может быть не оптимален для обучения большой языковой модели.

Следующий фактор — объём данных и требуемые вычислительные ресурсы.

Также важно учитывать уровень подготовки разработчиков. Для учебных задач полезен инструмент с понятным интерфейсом и хорошей документацией, тогда как исследовательский проект может требовать большей гибкости.

Необходимо учитывать возможность интеграции модели с существующими информационными системами.

Для реальных проектов существенное значение имеют стоимость, масштабируемость, сопровождаемость и безопасность.

Таким образом, выбор инструмента является частью проектирования интеллектуальной системы.

Безопасность при использовании инструментов искусственного интеллекта

Для магистрантов направления информационной безопасности этот вопрос особенно важен.

При использовании внешних AI-сервисов нельзя автоматически считать допустимой передачу любых данных.

Представим, что сотрудник организации загружает внутренний конфиденциальный документ в публичный AI-сервис для создания краткого содержания.

С технической точки зрения задача может быть выполнена успешно. Но с точки зрения информационной безопасности возникает вопрос: разрешено ли передавать данный документ внешней системе?

Аналогичная проблема возникает при работе с персональными данными, медицинской информацией, исходным кодом закрытых проектов и служебными документами.

Поэтому перед использованием внешней AI-платформы необходимо понимать, какие данные передаются сервису и какие правила обработки данных действуют в организации.

Другой риск связан с зависимостью от результатов модели. Ответ, созданный системой искусственного интеллекта, не должен автоматически восприниматься как достоверный.

Например, генеративная модель может сформировать программный код, содержащий ошибку или небезопасное решение. Поэтому результат должен проверяться специалистом.

Ваш силлабус прямо предусматривает безопасное и ответственное использование технологий искусственного интеллекта, включая вопросы этики, авторского права и защиты информации.

Инструменты ИИ как единая экосистема

Важно понимать, что в реальном проекте обычно используется не один инструмент, а их комбинация.

Рассмотрим пример системы обнаружения сетевых атак.

Исходный набор сетевого трафика может быть загружен и исследован в Jupyter Notebook. Для обработки таблицы применяется Pandas. Числовые операции выполняются средствами NumPy. Визуализация помогает исследовать распределение данных.

Затем с помощью scikit-learn можно создать базовую модель классификации.

Если требуется исследовать более сложную нейронную архитектуру, можно перейти к PyTorch или TensorFlow.

После обучения модель необходимо сохранить и интегрировать в программную систему, которая будет получать новые данные и формировать прогноз.

Таким образом, инструменты не конкурируют друг с другом во всех ситуациях. Они могут выполнять разные функции в рамках одного проекта.

Общую схему можно представить следующим образом:

Данные → Pandas/NumPy → Анализ и визуализация → scikit-learn или TensorFlow/PyTorch → Обученная модель → API/приложение → Пользователь

Именно понимание такой экосистемы позволяет перейти от простого запуска готовой модели к полноценной разработке интеллектуального программного решения.

Заключение

Инструменты и платформы искусственного интеллекта представляют собой программную основу для практической реализации методов ИИ. Они используются для подготовки и анализа данных, создания моделей, их обучения, оценки, развертывания и последующей эксплуатации.

Одним из основных языков разработки в области искусственного интеллекта является Python. Jupyter Notebook и облачные интерактивные среды позволяют организовать экспериментальную работу с кодом и данными. NumPy обеспечивает эффективные численные вычисления, а Pandas применяется для обработки структурированных данных.

Для классических алгоритмов машинного обучения широко используются специализированные библиотеки, такие как scikit-learn. Для глубокого обучения применяются фреймворки TensorFlow/Keras и PyTorch. Экосистемы предварительно обученных моделей позволяют использовать transfer learning и адаптировать готовые модели под конкретные задачи.

При этом выбор инструмента не должен определяться только его популярностью. Необходимо учитывать характер задачи, объём и тип данных, доступные вычислительные ресурсы, квалификацию разработчиков, возможности интеграции, стоимость и требования информационной безопасности.

Особенно важно помнить, что AI-инструмент не заменяет понимание методов искусственного интеллекта. Готовая платформа может автоматически обучить модель, однако специалист должен понимать, какие данные используются, каким образом оценивается качество модели, существуют ли ошибки и ограничения и насколько безопасно применение полученного решения.

Именно поэтому в последующих лекциях курса мы переходим от знакомства с инструментами к непосредственному изучению **машинного обучения, подготовки данных, классификации, регрессии, кластеризации, глубокого обучения, NLP, компьютерного зрения и генеративного искусственного интеллекта**.

Контрольные вопросы

1. Что понимается под инструментом искусственного интеллекта?
2. Что такое AI-платформа и чем она отличается от отдельной библиотеки?
3. Какие основные этапы включает жизненный цикл AI-проекта?
4. Почему Python широко используется в искусственном интеллекте?
5. Для чего применяется Jupyter Notebook?
6. В чём особенность облачных сред типа Google Colab?
7. Для каких задач используется NumPy?
8. Что представляет собой DataFrame и для чего применяется Pandas?
9. Почему необходимо исследовать и визуализировать данные до обучения модели?
10. Для каких задач применяется scikit-learn?
11. Что такое обучающая и тестовая выборки?
12. Для чего используются TensorFlow и Keras?
13. Каково назначение PyTorch?
14. Что называется предварительно обученной моделью?
15. В чём заключается идея transfer learning?
16. Что представляет собой API и как он используется в AI-приложениях?
17. Чем отличаются No-Code и Low-Code подходы?
18. Какие факторы необходимо учитывать при выборе AI-платформы?
19. Какие риски возникают при передаче данных внешним AI-сервисам?
20. Почему использование готового AI-инструмента не заменяет знания методов искусственного интеллекта?