Цифриум

Технологическая карта

Умный код на Python. Базовый уровень

Модуль 3 | Тема 4 | Урок 2

Понимание архитектуры GPT: принципы работы, важность данных

Уважаемый коллега, обратите внимание!

Учёт присутствия ребёнка на уроке ведётся только по цифровому следу. Каждый ученик из группы на уроке должен передать цифровой след на нашей платформе.

Вы отвечаете за передачу следа учеником на нашей платформе во время урока.

Передавать цифровой след нужно в личном кабинете ученика.

Цифровой след засчитывается по итогу выполнения учеником трёх задач на платформе в ходе урока.

Обращаем внимание — в задачах с отправкой ответа в виде файла наша платформа поддерживает следующие форматы: `.pdf, .doc, .png, .jpeg, .xlsx, .mp3, .py, .ipynb, .txt, .csv, .json, .xml, .sb3, .ino, .hex`.

Контролируйте время урока, чтобы не пропустить момент, когда ученики должны залогиниться в личном кабинете и перейти к решению задач на платформе.

Проследите, чтобы каждый ученик залогинился в личном кабинете и выполнил 3 задачи на платформе. В противном случае, Вам не будет засчитано, что вы провели урок, занятие не будет оплачено, а ученику не будет выставлена отметка о присутствии на уроке.

Просим вас в ходе 45 минут урока проконтролировать, чтобы:

1. Каждый из учеников залогинился в личном кабинете.
2. Каждый ученик в разделе **«Решить задания»** отправил решения по 3 задачам — «Практическое задание 1», «Практическое задание 2» и «Практическое задание 3».

О занятии

Краткое содержание занятия:

1. Узнаете, что такое трансформер и как он работает.
2. Поймёте, как работает механизм внимания (attention) и почему он важен.
3. Узнаете, зачем нужно позиционное кодирование.
4. Изучите этапы обучения GPT: pre-training и fine-tuning.
5. Поймёте, как качество данных влияет на работу модели.
6. Проведёте практический эксперимент с визуализацией внимания в BertViz.

Цель:

Познакомиться с внутренней структурой модели GPT и ролью качества данных.

Задачи:

1. Изучить архитектуру трансформера: механизмы внимания (Attention) и позиционное кодирование.
2. Понять этапы предобучения (pre-training) и дообучения (fine-tuning).
3. Осознать значимость объёма и чистоты данных для качества генерации.
4. Провести практический эксперимент: визуализировать внимание на примере BertViz, сравнить результаты при «чистых» и «шумных» данных.

Термины

1. **Искусственный интеллект (ИИ)**

Область информатики, занимающаяся созданием систем, способных выполнять задачи, традиционно требующие человеческого интеллекта: распознавание речи, принятие решений, обучение и планирование.

2. **Трансформер**

О занятии

Архитектура нейросети, которая анализирует последовательности (например, текст) с помощью механизма внимания, обрабатывая все элементы одновременно, а не последовательно.

3. **Attention (Внимание)**

Механизм, позволяющий модели учитывать связи между словами вне зависимости от расстояния между ними в тексте.

4. **Позиционное кодирование**

Способ, которым модель понимает порядок слов в предложении, добавляя информацию о позиции каждого токена.

5. **Pre-training (Предобучение)**

Этап обучения модели на большом массиве данных без конкретной задачи (например, предсказание следующего слова).

6. **Fine-tuning (Дообучение)**

Дообучение модели под конкретную задачу (например, чат, перевод, классификация текста).

7. **BertViz**

Инструмент для визуализации внутренних процессов внимания в моделях-трансформерах.

8. **GPT (Generative Pre-trained Transformer)**

Семейство языковых моделей, основанных на архитектуре трансформера, обученных генерировать текст.

Необходимые материалы:

1. Доступ к платформе для выполнения практических заданий

Темы и время

ЭТАП	ВРЕМЯ
Приветствие	5 мин.
Теория	25 мин.
Итоги занятия	5 мин.
Получение фидбека. Рефлексия. Проверка знаний. Домашнее задание	5 мин.
Вопрос-ответ	5 мин.
Итого	45 мин.

Ход занятия

Номер слайда	Пояснение к слайду
1	**Приветствие учеников**
2	**Титульный слайд** Тема нашего урока: «Понимание архитектуры GPT: как работает искусственный интеллект». Сегодня мы станем цифровыми детективами и заглянем внутрь «мозгов» нейросети, чтобы понять, как она читает, думает и отвечает нам. Вы узнаете, почему компьютер понимает человеческий язык и как он учится на наших текстах.
3	**На прошлом занятии вы:** Давайте быстро освежим память! На прошлом уроке мы говорили о том, что такое искусственный интеллект (ИИ) — это программы, которые умеют учиться, как люди. Мы вспоминали, что раньше ИИ учили правилам вручную (символический подход), а теперь он учится сам на примерах (машинное обучение). Также мы касались темы нейросетей — это как слои нейронов в нашем мозге, только в компьютере.
4	**Сегодня на занятии вы:** Сегодня мы разберем главные секреты GPT. Мы узнаем, что такое трансформер — «двигатель» современных нейросетей. Поймем, как работает механизм внимания — способность модели выделять главное. Разберемся, почему порядок слов важен (позиционное кодирование). Узнаем, как учат модель: от простого чтения книг до сложных задач. И самое интересное — проведем эксперимент в BertViz и своими глазами увидим, на что «смотрит» нейросеть, когда пишет текст..
5	**Цель и задачи урока** Наша цель: не просто пользоваться чат-ботом, а понять, как он устроен внутри, и почему качество информации так важно. Наши задачи: Разобраться в архитектуре трансформера простыми словами. Понять, из каких этапов состоит обучение ИИ.

Ход занятия

	Осознать, почему плохие данные портят даже умную модель. Провести практический опыт с визуализацией внимания.
6	**Что такое трансформер?** Представьте, что вы читаете книгу. Обычно вы читаете слово за словом, слева направо. Старые нейросети делали так же. Но трансформер — это как супер-скоростной читатель: он «видит» весь текст сразу, целиком! Его главная фишка — механизм внимания (attention). Это способность модели мгновенно определять, какие слова в предложении связаны друг с другом, даже если они стоят далеко. Именно это делает современные переводчики и чат-боты такими умными.
7	**Как работает внимание (attention)?** Давайте разберем на примере: «Кошка испугалась собаки, потому что она громко залаяла». Вопрос на засыпку: кто залаял? Кошка или собака? Конечно, собака. Но чтобы это понять, нужно связать слово «она» со словом «собака». Механизм внимания работает как маркер: он проводит невидимую связь от местоимения к нужному слову. Модель взвешивает все слова в предложении и понимает: в данном контексте «она» — это собака. Без этого механизма компьютер бы просто запутался.
8	**Почему важны позиции слов?** Трансформер видит все слова сразу, как рассыпанные на столе карточки. Но в языке порядок решает всё! Сравните: «Кот гонит собаку» и «Собаку гонит кот». Слова одни и те же, а смысл противоположный. Чтобы модель не перепутала, кто за кем бежит, изобрели позиционное кодирование. Это специальные метки, которые говорят модели: «это слово первое», «это второе» и так далее. Без них нейросеть не смогла бы отличить вопрос от ответа.
9	**Как обучают GPT?** Обучение модели похоже на школу и университет. Этап 1: Pre-training (Базовое обучение). Это как начальная школа, но в огромном масштабе. Модель читает миллиарды текстов из интернета и учится предсказывать следующее слово в предложении. Она просто запоминает

Ход занятия

	закономерности: после слова «желтый» часто идет «лимон» или «свет». Этап 2: Fine-tuning (Тонкая настройка). Это как выбор профессии. Уже умную модель учат конкретной задаче: отвечать на вопросы, писать код или поддерживать беседу. Здесь ей помогают учителя-люди, которые показывают, как нужно отвечать правильно.
10	**Что влияет на обучение?** Нейросеть учится на том, что мы ей даем. Хорошие данные — это как качественные учебники: грамотные тексты, интересные книги, проверенные факты. На них модель учится говорить красиво и по делу. Плохие данные — это как испорченный учебник: опечатки, грубости, ложная информация, бессвязный набор слов. Есть золотое правило информатики: GIGO (Garbage In, Garbage Out) — «Мусор на входе — мусор на выходе». Если кормить модель мусором, она и в ответ выдаст мусор, какой бы умной она ни была.
11	**Практика: смотрим глазами GPT** А теперь эксперимент! Мы используем программу BertViz, которая рисует «карту мыслей» нейросети. План действий: Введем чистое предложение: «Кот подошёл к собаке, и она зарычала». Нажмем на слово «она» и посмотрим, с каким словом оно соединится линией. Затем добавим в текст «шум» — опечатки и перестановки: «Кот подошёл к собаке. Она... зарычала, к подошёл кот». Снова посмотрим на линии внимания и сравним результат. (Алгоритм выполнения описан в примере выполнения задания 1)
12	**Что показывает эксперимент?** Что мы увидим? В первом случае, с чистым текстом, линия от слова «она» будет уверенно тянуться к слову «собаке». Модель всё поняла правильно. Во втором случае, с «шумом», линии начнут путаться, расползаться в разные стороны или указывать на случайные слова. Модель сбивается с толку!

Ход занятия

	Вывод: даже умный алгоритм зависит от качества ввода. Четкий текст = четкий ответ. Запутанный текст = ошибка в логике. Подумайте: почему одна пропущенная запятая или опечатка может заставить ИИ выдать совершенно неверный смысл?
13	**Подведение итогов занятия** Давайте закрепим! Сегодня мы узнали, что GPT — это не магия, а сложная архитектура трансформера. Мы поняли, что механизм внимания помогает связывать слова, а позиционное кодирование сохраняет порядок. Мы увидели, что обучение идет в два этапа, и что данные — это самое главное «топливо» для ИИ. Вопрос вам на дом: что вас удивило больше всего? Может быть, то, как модель ищет связи между словами, или то, насколько она чувствительна к ошибкам?

Практика

Вопросы для самопроверки

- Как работает механизм внимания (attention)?
- Зачем нужно позиционное кодирование?
- В чём разница между pre-training и fine-tuning?
- Почему порядок слов влияет на смысл?
- Как влияет «шум» на поведение языковой модели?

Задание 1

- Запустить среду с моделью 'bert-base-uncased' и библиотекой bertviz.
- Ввести предложение: «Кот подошёл к собаке, и она зарычала».
- Проанализировать, на какое слово направлено внимание от токена «она».
- Изменить на искажённое предложение: «Кот подошёл к собаке. Она... зарычала, к подошёл кот».
- Повторить визуализацию и сравнить с исходной.

Задание 2.

- Что изменила структура фразы?
- Кого теперь модель считает «она»?
- Повлияло ли добавление «шума» на работу attention?

Примеры выполнения заданий

Проверь себя по следующим критериям:

- Механизм внимания (attention) — позволяет модели «фокусироваться» на наиболее релевантных частях входного текста при генерации каждого слова, взвешивая их важность.
- Позиционное кодирование — добавляет информацию о порядке слов в последовательности, так как трансформеры сами по себе не учитывают позицию (в отличие от RNN).
- Pre-training vs fine-tuning:
- Pre-training — обучение модели на огромном общем корпусе текста для усвоения языковых закономерностей.
- Fine-tuning — дообучение предобученной модели на специализированных данных для конкретной задачи (например, классификация или ответы на вопросы).
- Порядок слов влияет на смысл, потому что язык — структурированная система: изменение порядка может менять субъект/объект, причинно-следственные связи или акцент (напр., «Кот гонится за мышью» ≠ «Мышь гонится за котом»).
- «Шум» (ошибки, бессмысленные токены, искажения во входе) может дезориентировать модель, снижая качество вывода — хотя современные модели частично устойчивы к шуму благодаря обучению на разнообразных данных.

Задание 1

1) Откройте Google Colab (нужен аккаунт Google) и создайте **Новый блокнот**.

2) Вставьте весь этот код в первую ячейку и нажмите кнопку "Выполнить"

```
# 1. Установка библиотек

!pip install bertviz transformers matplotlib -q


# 2. Импорт библиотек

import torch
```

Примеры выполнения заданий

```
import matplotlib.pyplot as plt

from transformers import BertTokenizer, BertModel

from bertviz import head_view

import warnings

warnings.filterwarnings('ignore')


# 3. Загрузка модели

print("Загрузка модели...")

model_name = "bert-base-uncased"

tokenizer = BertTokenizer.from_pretrained(model_name)

model = BertModel.from_pretrained(model_name, output_attentions=True)

print("Модель загружена!\n")


# 4. Подготовка предложений

text_clean = "The cat approached the dog, and it growled"

text_noisy = "The cat approached the dog. It... growled, approached cat the"


print("="*60)

print("ЧИСТЫЙ ТЕКСТ:  The cat approached the dog, and IT growled")

print("ШУМНЫЙ ТЕКСТ:   The cat approached the dog. IT... growled,
approached cat the")
```

Примеры выполнения заданий

```
print("="*60)

print("\n👉 Сейчас откроются ДВЕ визуализации от bertviz")

print("👉 Сравните их и увидите разницу!\n")


# 5. Функция для визуализации

def показать_bertviz(текст, номер):

    print(f"\n--- Визуализация {номер} ---")

    inputs = tokenizer(текст, return_tensors='pt')


    with torch.no_grad():

    outputs = model(**inputs)


    attention = outputs.attentions

    tokens = tokenizer.convert_ids_to_tokens(inputs['input_ids'][0])


    print(f"Токены: {tokens}")

    print("🔍 Найдите слово 'it' и посмотрите на его связи")


    # Показываем визуализацию bertviz

    head_view(attention, tokens)
```

Примеры выполнения заданий

```
# 6. Показываем первую визуализацию

показать_bertviz(text_clean, "1 (ЧИСТЫЙ ТЕКСТ)")


input("\n⏸️ Нажмите Enter для второй визуализации...")


# 7. Показываем вторую визуализацию

показать_bertviz(text_noisy, "2 (ШУМНЫЙ ТЕКСТ)")


# 8. Создаем простое пояснение с графиком

print("\n" + "="*60)

print("📊 СРАВНЕНИЕ РЕЗУЛЬТАТОВ")

print("="*60)


# Получаем числовые значения для графика

def получить_числа(текст):

        inputs = tokenizer(текст, return_tensors='pt')

        tokens = tokenizer.convert_ids_to_tokens(inputs['input_ids'][0])


        with torch.no_grad():

        outputs = model(**inputs)
```

Примеры выполнения заданий

```
        # Берем средний слой для наглядности
        attention = outputs.attentions[5][0, 8].numpy()

        it_idx = None
        dog_idx = None

        for i, token in enumerate(tokens):
        if token == 'it':
        it_idx = i
        elif token == 'dog':
        dog_idx = i

        if it_idx is not None and dog_idx is not None:
        return attention[it_idx][dog_idx]
        return 0

# Собираем данные
clean_value = получить_числа(text_clean)
noisy_value = получить_числа(text_noisy)

# Рисуем простой график
```

Примеры выполнения заданий

```
plt.figure(figsize=(8, 5))

bars = plt.bar(['ЧИСТЫЙ ТЕКСТ', 'ШУМНЫЙ ТЕКСТ'], [clean_value,
noisy_value],

        color=['green', 'red'])

plt.ylabel('Сила связи "it" → "dog"', fontsize=12)

plt.title('Сравнение понимания моделью слова "it"', fontsize=14)

plt.ylim(0, 0.6)


# Добавляем значения на столбцы

for bar in bars:

        height = bar.get_height()

        plt.text(bar.get_x() + bar.get_width()/2., height,

        f'{height:.3f}', ha='center', va='bottom', fontsize=12)


plt.axhline(y=0.3, color='blue', linestyle='--', alpha=0.7, label='Порог
понимания')

plt.legend()

plt.grid(axis='y', alpha=0.3)

plt.show()


# 9. Итоговый вывод

print("\n" + "="*60)
```

Примеры выполнения заданий

print("🔍 ЧТО ВЫ УВИДЕЛИ В BERTVIZ:")

print("="*60)

print("""

В ПЕРВОЙ визуализации (чистый текст):

• Найдите слово 'it' (в районе 8-9 позиции)

• От него идут яркие линии

• Одна из самых ярких линий ведет к слову 'dog'

✅ Модель ПОНЯЛА, что 'it' = собака

ВО ВТОРОЙ визуализации (шумный текст):

• Снова найдите слово 'it'

• Линии от него стали бледнее и размазаны

• Связь с 'dog' почти исчезла или очень слабая

❌ Модель ЗАПУТАЛАСЬ

📊 ГРАФИК ПОКАЗЫВАЕТ ТО ЖЕ САМОЕ:

Чистый текст: связь = {:.3f} (ВЫСОКАЯ)

Шумный текст: связь = {:.3f} (НИЗКАЯ)

ВЫВОД: Шум в тексте ломает способность модели понимать смысл!

""".format(clean_value, noisy_value))

Примеры выполнения заданий

В чистом тексте (левый график):

Высокие столбцы у слов:

- , (запятая) - потому что после запятой всегда идет "and"
- and - потому что "and" соединяет части предложения
- growled - потому что "it" (она) выполняет действие "growled" (зарычала)

Это как в русском языке: когда мы говорим "она зарычала", слово "она" тесно связано и с существительным (кто?), и с глаголом (что сделала?).

В шумном тексте (правый график):

Высокие столбцы у слов:

- . (точка) - появился разрыв
- ... (многоточие) - мусорный токен
- cat - модель путается между "cat" и "dog"

Задание 2

- Структура фразы стала нелогичной и грамматически неправильной. Нарушен естественный порядок слов, появились паузы (многоточие), что затрудняет понимание контекста.
- Из-за шума часть внимания может перейти к слову «кот» (последнему упомянутому существительному) или даже к «зарычала». То есть модель ошибочно связывает «она» с ближайшим в тексте существительным, а не с «собакой».
- Да. Карты внимания стали более размытыми: важность ключевого слова («собака») снизилась, а веса распределились по дополнительным токенам. Это демонстрирует, что «шум» ухудшает фокусировку внимания на релевантных элементах фразы.

Итоги занятия

Сегодня на занятии вы:

- Познакомились с внутренней структурой модели GPT.
- Изучили архитектуру трансформера и механизмы внимания (Attention).
- Поняли роль позиционного кодирования для понимания порядка слов.
- Узнали этапы предобучения (pre-training) и дообучения (fine-tuning).
- Осознали значимость объёма и чистоты данных для качества генерации.
- Провели практический эксперимент с BertViz и сравнили результаты при «чистых» и «шумных» данных..

На следующем занятии вы:

- Узнаете, как работать с API GPT и отправлять запросы к языковым моделям.
- Изучите формат JSON-запроса: поля model, messages, параметры генерации..
- Узнаете, какие параметры влияют на генерацию текста (temperature, max_tokens и др.)
- Научитесь подготавливать данные для эффективной работы с API.
- Будете отправлять запросы к GPT, настраивать параметры генерации и анализировать результаты.

Получение фидбека. Рефлексия. Проверка знаний

Попросите ребят решить интерактивные задачи по теме урока.

Ответы к интерактивным задачам

Практическое задание 1

Как называется механизм в трансформере, который позволяет модели учитывать связи между словами независимо от расстояния между ними?

внимание

Практическое задание 2

Что такое pre-training (предобучение) в контексте языковых моделей?

- Обучение модели на конкретной задаче, например, переводе текста.
- **Обучение модели на большом массиве данных для предсказания следующего слова.**
- Настройка параметров модели вручную программистом.
- Тестирование модели перед запуском в продакшен.

Практическое задание 3

Какие факторы влияют на качество работы языковой модели? (выберите все правильные варианты)

- **Объём данных для обучения**
- **Чистота и качество текстов в обучающей выборке**
- Цвет интерфейса программы
- **Наличие позиционного кодирования для учёта порядка слов**

Ответы к интерактивным задачам

Вопрос-ответ

Если во время подготовки или проведения урока возникают вопросы, их необходимо адресовать в методический чат.