Шпаргалка по трансформерам / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

April 2, 2025

Contents

1 Введение в Трансформеры (Attention, Architecture, LLMs)

1.1 Механизм Внимания (Attention)

Основная Идея Attention

Представьте, что вы переводите длинное предложение. Когда вы пишете очередное слово перевода, вы не смотрите на *все* слова исходного предложения одинаково. Вы обращаете **внимание** на одно или несколько конкретных слов оригинала, которые наиболее важны для перевода *именно этого* слова. Механизм **Attention** (Внимание) в нейронных сетях делает то же самое: он позволяет модели динамически фокусироваться на наиболее релевантных частях входной последовательности при генерации каждой части выходной последовательности. Технически, Attention вычисляет **веса (weights)** важности для каждого элемента входа относительно текущей задачи, показывая, "насколько" нужно обратить внимание на тот или иной элемент.
Ключевая цель: Дать модели понять, какие части входа наиболее важны для текущего шага обработки.

Аналогия: Повар и Рецепт

Представьте, что вы готовите сложное блюдо по рецепту (входная последовательность). На каждом шаге (генерация действия) вы не перечитываете весь рецепт заново. Вы **обращаете внимание** на конкретный пункт или ингредиент, релевантный для *текущего* действия (например, "добавить 2 яйца", "взбивать 5 минут"). Механизм Attention работает похожим образом.

Self-Attention (Внимание к Себе)

Это особый вид Attention, где модель взвешивает важность **разных слов внутри одной и той же последовательности** по отношению друг к другу.

- Позволяет модели понять внутренние зависимости в предложении. Например, в предложении "Банк выдал кредит, потому что **он** был одобрен", Self-Attention помогает понять, что местоимение "**он**" относится к слову "кредит", а не "Банк".
- Является ключевым компонентом архитектуры **Transformer**.

Суть: Каждое слово "смотрит" на все остальные слова в предложении (включая себя) и решает, насколько они важны для понимания его собственного контекста.

1.2 Архитектура Трансформера (Общая Схема)

Encoder-Decoder Структура и Ключевые Элементы

Классическая архитектура **Transformer** часто состоит из двух основных частей:

- **Encoder (Кодировщик):** Читает всю входную последовательность (например, предложение на русском) и создает ее богатое контекстуальное представление. Он использует **Self-Attention**, чтобы понять взаимосвязи между словами во входных данных.
- **Decoder (Декодировщик):** Генерирует выходную последовательность (например, перевод на английский) шаг за шагом. На каждом шаге он использует:
  - **Self-Attention** для учета уже сгенерированных им слов.
  - **Encoder-Decoder Attention** для фокусировки на релевантных частях представления, полученного от кодировщика.

Ключевые "строительные блоки" внутри Encoder и Decoder:

- **Multi-Head Attention** (позволяет механизму внимания одновременно фокусироваться на разных типах зависимостей или 'аспектах' информации, как если бы у вас было несколько 'голов', читающих текст с разными целями).
- **Feed-Forward Networks** (обычные полносвязные сети, применяемые к каждому элементу последовательности независимо).
- **Positional Encoding (Позиционное Кодирование): Критически важно!** Сам по себе Self-Attention не учитывает порядок слов (для него "собака укусила человека" и "человека укусила собака" – одно и то же, так как он смотрит на все слова сразу). Positional Encoding решает эту проблему, добавляя к эмбеддингу каждого слова специальный вектор, содержащий информацию о его *позиции* в последовательности. **Аналогия:** Как номер страницы в книге помогает понять порядок глав.

Примечание: Существуют модели, использующие только Encoder (как BERT) или только Decoder (как GPT).

Аналогия: Переводчик-Человек

- **Encoder:** Опытный переводчик читает исходное предложение целиком, вникает в смысл и взаимосвязи слов (Self-Attention), учитывая их порядок (Positional Encoding). Он формирует у себя в голове полное понимание ("контекстуальное представление").
- **Decoder:** Переводчик начинает писать перевод. Для каждого слова перевода он вспоминает, что уже написал (Decoder Self-Attention + Positional Encoding) и обращается к своему пониманию оригинала, фокусируясь на нужных словах (Encoder-Decoder Attention).

1.3 Примеры: BERT и GPT (Большие Языковые Модели - LLM)

Большие Языковые Модели (LLM - Large Language Models)

Это модели (часто основанные на архитектуре Transformer), обученные на **огромных** объемах текстовых данных. Они способны понимать и генерировать человеческий язык с высокой степенью точности.

- **BERT (Bidirectional Encoder Representations from Transformers):**
  - Использует в основном **Encoder** часть Трансформера.
  - Обучается предсказывать пропущенные ("замаскированные") слова в тексте (**Masked Language Model - MLM**), а также предсказывать, является ли одно предложение логическим продолжением другого. Это позволяет ему учитывать контекст **с обеих сторон** от слова (Bidirectional), что дает глубокое понимание.
  - Отлично подходит для задач, требующих глубокого понимания контекста: классификация текста, извлечение именованных сущностей (NER), ответы на вопросы.
  - **Аналогия BERT:** Студент, которому дали текст с пропусками, и он должен их заполнить, видя весь остальной текст, чтобы доказать свое понимание.
- **GPT (Generative Pre-trained Transformer):**
  - Использует в основном **Decoder** часть Трансформера.
  - Обучается предсказывать **следующее слово** в последовательности (**Causal Language Model - CLM**), опираясь только на *предшествующий* контекст. Это делает модель **авторегрессионной** (генерирует слово за словом).
  - Отлично подходит для генеративных задач: написание текстов, создание диалоговых систем, продолжение кода.
  - **Аналогия GPT:** Опытный рассказчик, который может продолжить любую начатую историю, предсказывая наиболее вероятное следующее слово на основе уже сказанного.

Ключевое для Собеседования

Главное: Понимать концепцию **Attention** как механизма вычисления весов важности частей входа. Знать, что **Self-Attention** смотрит на зависимости внутри одной последовательности, но **не видит порядок** слов, поэтому необходимо **Positional Encoding**. Представлять **Transformer** как архитектуру (часто Encoder-Decoder) с **Multi-Head Attention** и **Feed-Forward** блоками. Знать, что **BERT** (Encoder, MLM, Bidirectional) и **GPT** (Decoder, CLM, Autoregressive) - это примеры **LLM** на базе Трансформеров с разными архитектурными фокусами и задачами предобучения, определяющими их сильные стороны (понимание контекста vs генерация).