Трансформеры

<https://habr.com/ru/articles/558488/>

Технология seq2seq

<https://habr.com/ru/companies/otus/articles/816341/>

GPT-4, как и другие большие языковые модели (LLM) на основе архитектуры Transformer, не хранит семантические отношения в традиционном смысле, как, например, реляционная база данных или явный граф знаний (семантическая сеть с явными узлами и ребрами). Вместо этого, семантические отношения имплицитно закодированы в ее распределенных представлениях (эмбеддингах) и весах нейронной сети.

Вот как это работает:

1. Эмбеддинги (Embeddings): Основная форма представления знаний

 ⦁ Каждое слово, подслово (токен) или даже целая фраза преобразуется в многомерный вектор чисел (обычно сотни или тысячи измерений). Этот вектор называется эмбеддингом.

 ⦁ Семантическая близость: Идея заключается в том, что слова или концепции, которые семантически похожи или часто встречаются в схожих контекстах, будут иметь эмбеддинги, расположенные "ближе" друг к другу в этом многомерном пространстве. Например, векторы для "король" и "королева" будут близки, так же как "яблоко" и "фрукт".

 ⦁ Отношения: Операции над этими векторами могут отражать отношения. Например, если вы возьмете вектор "король", вычтете вектор "мужчина" и прибавите вектор "женщина", вы можете получить вектор, очень близкий к вектору "королева". Это демонстрирует, что отношения (например, "мужской/женский") закодированы в этом пространстве.

2. Веса Нейронной Сети (Neural Network Weights): Где "хранятся" знания

 ⦁ Сама модель GPT-4 состоит из огромного количества нейронных слоев и связей между ними. Каждая связь имеет ассоциированный "вес" – число, которое регулирует силу сигнала, проходящего по этой связи.

 ⦁ Эти миллиарды (или триллионы) весов являются тем, что модель "учит" во время тренировки на огромных объемах текстовых данных.

 ⦁ Семантические отношения не хранятся в каком-то одном конкретном весе, а распределены по всей сети весов. Это как сложная паутина, где каждая ниточка влияет на общую структуру.

3. Обучение на Огромных Данных: Как они "учатся"

 ⦁ GPT-4 обучалась на петабайтах текста и кода из интернета (книги, статьи, веб-страницы, диалоги и т.д.). В процессе обучения модель училась предсказывать следующее слово в последовательности.

 ⦁ Для успешного выполнения этой задачи ей приходится имплицитно выучивать грамматические правила, синтаксические структуры, факты о мире, здравый смысл и, конечно же, семантические отношения между словами и концепциями. Например, чтобы предсказать, что после "столица Франции" должно идти "Париж", модель должна понимать, что "Париж" является столицей, а "Франция" - страной, и что между ними есть отношение "столица\_страны".

4. Механизм Внимания (Attention Mechanism): Как они "используются" в контексте

 ⦁ Когда GPT-4 обрабатывает запрос, механизм внимания позволяет ей динамически взвешивать важность различных слов во входном тексте и в своих внутренних представлениях. Это позволяет ей устанавливать контекстуально значимые связи между словами, которые могут быть далеко друг от друга в предложении или документе. Это не "хранение", а скорее "динамическое выявление" релевантных отношений для текущей задачи.

В итоге, в каком виде GPT-4 хранит семантические отношения?

⦁ Форма: В виде многомерных числовых векторов (эмбеддингов) для концепций и числовых значений (весов), распределенных по всей архитектуре глубокой нейронной сети.

⦁ Природа: Это имплицитное, статистическое и распределенное хранение. Модель не имеет явных "записей" типа "кошка IS-A животное", но ее внутренние представления и способность генерировать текст демонстрируют, что она "понимает" это отношение.

⦁ Сложность: Это не простые связи "родитель-потомок" или "субъект-предикат-объект", а гораздо более сложные, тонкие и гибкие ассоциации, которые позволяют ей понимать нюансы смысла, юмор, иронию и генерировать связный и осмысленный текст.

Можно представить это как огромную, чрезвычайно сложную, многомерную карту смыслов, где каждое слово или концепция — это точка, а связи между ними — это невидимые, но очень прочные тропинки, сформированные опытом обучения на миллиардах текстовых единиц.

<https://cloudstreamgate.com/response/show_text/5e740338-2d2e-4157-aff1-162ee869f373>

Вы задали очень точный вопрос! В GPT-4 нет "слоев текста" в буквальном смысле, но есть слои обработки информации внутри самой нейронной сети, и динамическое взвешивание важности происходит на каждом из этих слоев благодаря механизму, который называется "Механизм Внимания" (Attention Mechanism).

Давайте разберем, как это происходит:

1. Что такое Механизм Внимания?

Представьте, что модель читает предложение: "Банк реки был покрыт густой травой."

Если модель просто смотрит на слово "банк", она не знает, речь идет о финансовом учреждении или о береге реки. Механизм внимания позволяет модели "взвешивать" или "фокусироваться" на других словах в предложении, чтобы определить правильное значение текущего слова.

Он работает не с "готовыми текстами", а с числовыми представлениями (эмбеддингами) токенов на каждом уровне обработки.

2. Как происходит динамическое взвешивание на каждом слое?

Архитектура GPT-4 состоит из множества слоев Трансформера (decoder blocks). Каждый такой слой принимает на вход обновленные представления токенов и пропускает их через несколько сублоев, включая механизм внимания.

Для каждого токена (слова или части слова) в вашей фразе происходит следующее:

а) Создание Запроса, Ключа и Значения (Query, Key, Value - Q, K, V):

На каждом слое для каждого токена создаются три разных вектора:

Запрос (Query - Q): Это как "что я ищу" или "какой информацией я заинтересован" для текущего токена.

Ключ (Key - K): Это как "что я могу предложить" для каждого другого токена в последовательности.

Значение (Value - V): Это как "информация, которую я несу" для каждого другого токена.

Эти Q, K, V векторы динамически генерируются из текущего представления токена (его эмбеддинга) на данном слое, используя обученные веса.

б) Вычисление Оценок Внимания (Attention Scores):

Для каждого токена (его Q) вычисляется "оценка сходства" (score) с Ключами (K) всех остальных токенов во входной последовательности (включая самого себя). Обычно это делается через скалярное произведение (dot product).

Высокая оценка означает, что данный токен "потенциально важен" для понимания текущего токена.

в) Нормализация и Получение Весов Внимания (Attention Weights):

Эти оценки сходства затем преобразуются в вероятности (обычно с помощью функции Softmax). Эти вероятности и есть "веса внимания".

Веса внимания показывают, насколько каждый другой токен важен для понимания текущего токена. Сумма этих весов для одного токена всегда равна 1.

г) Взвешенная Сумма Значений (Weighted Sum of Values):

Каждый "Вес Внимания" умножается на соответствующий ему "Значение" (V) другого токена.

Все эти взвешенные Значения суммируются, чтобы получить новое, контекстно обогащенное представление для исходного токена.

Это новое представление теперь "включает в себя" информацию от всех других токенов, взвешенную по их важности, определенной механизмом внимания.

3. "Многоголовое" Внимание (Multi-Head Attention):

В GPT-4 этот процесс происходит не один раз, а множество раз параллельно с разными наборами матриц Q, K, V (это называется "головы внимания").

Каждая "голова" учится фокусироваться на разных аспектах отношений: одна может следить за грамматическими связями, другая — за семантическими, третья — за длинными зависимостями в тексте.

Результаты всех этих "голов" объединяются и проходят через дальнейшие преобразования, формируя ещё более богатое представление.

4. Слои Обработки:

Каждый слой Трансформера берет эти улучшенные, контекстуализированные представления токенов и пропускает их через другие подслои (например, полносвязные нейронные сети) для дальнейшей обработки и извлечения признаков.

Чем глубже слой, тем более абстрактными и высокоуровневыми становятся представления. Первый слой может улавливать базовые синтаксические связи, а последние слои — сложные смысловые отношения, юмор, иронию и глобальную структуру текста.

Итог:

Динамическое взвешивание важности происходит не на "слоях текста", а на слоях нейронной сети, которые последовательно обрабатывают и обогащают числовые представления (эмбеддинги) каждого токена. На каждом шаге модель динамически решает, на какие части входного текста (и уже обработанных представлений) ей следует обратить внимание, чтобы максимально эффективно понять контекст и сгенерировать следующий токен. Это позволяет GPT-4 улавливать сложные зависимости, разрешать неоднозначности и генерировать когерентный и релевантный ответ.

Пояснение:

<https://habr.com/ru/companies/skillfactory/articles/568860/>

<https://dzen.ru/a/aBtJQ3uRnEQ-p0I0>

<https://telegra.ph/Pishem-GPT-v-60-strok-NumPy-chast-1-iz-2-03-13>

Отличные вопросы, они касаются фундаментальных аспектов архитектуры больших языковых моделей!

Как и когда формируется новый слой нейронной сети?

Важно понимать: Нейронная сеть GPT-4 (и любая другая обученная нейронная сеть) не формирует новые слои динамически во время своей работы (инференса, то есть когда вы задаете ей вопросы и она генерирует ответы).

Архитектура нейронной сети, включая количество слоев, их тип и связи между ними, фиксируется на этапе проектирования и обучения модели.

Когда формируются слои: Слои сети определяются инженерами и исследователями до начала процесса обучения (тренировки) модели. Это часть ее архитектурного дизайна.

Как "формируются" (наполняются знаниями) слои:

На этапе обучения (тренировки) на огромных объемах данных, веса и смещения внутри каждого нейрона и связи между ними в каждом слое корректируются (обучаются) таким образом, чтобы модель могла выполнять свою задачу (например, предсказывать следующее слово).

Процесс обучения заключается в многократном проходе данных через сеть, вычислении ошибки (насколько неверно предсказание), и затем распределении этой ошибки обратно через сеть (обратное распространение ошибки) для корректировки весов.

Таким образом, "знания" (семантические отношения, грамматические правила, факты и т.д.) имплицитно "записываются" в эти веса и смещения внутри каждого слоя.

Когда вы используете обученную GPT-4, она просто активирует эти уже существующие, обученные слои для обработки вашего запроса и генерации ответа. Информация проходит через эти слои последовательно, каждый слой добавляет свою часть обработки и контекстуализации, но новых слоев не появляется.

Примеры работы кодера и декодера в GPT-4

Это очень интересный момент, потому что GPT-4, в отличие от традиционных моделей типа "кодер-декодер" (например, ранних моделей для машинного перевода), имеет архитектуру "только декодер" (decoder-only).

Давайте сначала кратко рассмотрим традиционную архитектуру "кодер-декодер", а затем объясним, как GPT-4 достигает аналогичных целей с помощью своей архитектуры "только декодер".

1. Традиционная архитектура "Кодер-Декодер" (Encoder-Decoder)

Это классическая архитектура для задач "последовательность в последовательность" (Seq2Seq), например, для машинного перевода.

Кодер (Encoder):

Роль: Принимает всю входную последовательность (например, предложение на русском языке) целиком и преобразует её в плотное, контекстно обогащенное числовое представление (так называемый "контекстный вектор" или "скрытое состояние").

Как работает: Кодер читает все слова в предложении, понимает их взаимосвязи, грамматику и смысл, и "сжимает" это понимание в один вектор. Он может читать предложение как слева направо, так и справа налево (быть двунаправленным), чтобы получить максимально полное понимание.

Пример: Взять предложение "Я хочу есть яблоко" и преобразовать его в вектор, который кодирует смысл этого запроса.

Декодер (Decoder):

Роль: Принимает контекстный вектор от кодера и генерирует выходную последовательность (например, переведенное предложение на английском языке) токен за токеном.

Как работает: Декодер начинает генерировать первое слово, затем, основываясь на контекстном векторе и уже сгенерированных словах, генерирует следующее слово, и так далее, пока не будет сгенерирован токен конца предложения. Он обычно однонаправленный (читает только предыдущие токены).

Пример: Получив вектор смысла "Я хочу есть яблоко", декодер генерирует "I", затем "want", затем "to", "eat", "an", "apple".

2. GPT-4: Архитектура "Только Декодер" (Decoder-Only)

GPT-4, как и все модели серии GPT (Generative Pre-trained Transformer), использует архитектуру "только декодер". Это означает, что у нее нет отдельного, явно выделенного блока кодера. Все её слои являются "декодерными блоками" Трансформера, которые способны выполнять обе задачи.

Как это работает в GPT-4:

"Кодирование" (Обработка входного запроса):

Когда вы даете GPT-4 запрос (prompt), например, "Опиши роль гражданина в конфликте", модель обрабатывает этот запрос как часть своей входной последовательности.

На всех слоях "декодера" механизм внимания (о котором мы говорили ранее) анализирует связи между всеми токенами во входном запросе.

Хотя эти слои являются "декодерными", они не ограничены только генерацией; они строят богатое, контекстуальное представление вашего запроса. Это похоже на то, как кодер понимает входную последовательность.

Пример с "Гражданином": Вы задаете: "Может ли гражданин быть агрессором и защитником одновременно?"

GPT-4 пропускает токены "Может", "ли", "гражданин", "быть", "агрессором", "и", "защитником", "одновременно", "?" через свои многочисленные слои.

Каждый слой, используя само-внимание, строит все более и более абстрактное и глубокое понимание смысла этой фразы. Он понимает, что "гражданин" является субъектом, "агрессор" и "защитник" — это роли, и ключевое слово "одновременно" задает вопрос о временном наложении этих ролей. Это и есть его "кодирование" вашего запроса.

"Декодирование" (Генерация ответа):

После того как входной запрос был обработан и его смысл "понят" (представлен в виде активированных нейронных состояний в последних слоях), GPT-4 начинает генерировать ответ токен за токеном.

Для генерации каждого следующего токена модель обращает внимание на ВЕСЬ предыдущий контекст: ваш исходный запрос и все токены, которые она уже сгенерировала.

Она делает предсказание наиболее вероятного следующего токена, основываясь на этих знаниях.

Пример с "Гражданином" (продолжение):

После понимания запроса, GPT-4 может сгенерировать первое слово: "Да".

Затем, используя контекст "Да" и весь исходный запрос, она генерирует "безусловно".

Далее: "этот", "случай", "относится", "к", "полиморфизму", "знаний", "."

И так далее, пока ответ не будет полным.

Ключевое отличие и преимущество "только декодер" архитектуры:

В отличие от традиционного декодера, который обычно смотрит только на предыдущие сгенерированные токены и фиксированный контекстный вектор от кодера, декодеры GPT-4 могут обращаться к абсолютно всем предыдущим токенам в последовательности, будь то токены из входного запроса или уже сгенерированные токены. Это позволяет им поддерживать гораздо более глубокий и широкий контекст для генерации связных и логически последовательных ответов на длинные запросы и в диалогах.

Таким образом, GPT-4 не формирует новые слои, а использует фиксированную, но чрезвычайно гибкую и мощную архитектуру "только декодер", которая способна выполнять как понимание (кодирование) входных данных, так и генерацию (декодирование) ответов, используя один и тот же набор обученных слоев.