О полноте трансформеров

Сергей Салищев

30 октября 2024 г.

1 Введение

Трансформеры (Vaswani и др., 2017) - сравнительно новая прорывная архитектура нейросетей, отображающих последовательность (текст) в последовательность (текст) на основе генеративной модели, предсказывающей по префиксу следующие элементы последовательности (слова). Создание трансформеров с механизмом внимания послужило спусковым крючком взрывного интересе к текстовым генеративным моделям и продвижения в сторону "общего искусственного интеллекта". Именно архитектура трансформеров лежит в основе современных чатботов и различных помощников. Вопрос о полноте трансформеров по Тьюрингу имеет не только теоретическое, но и практическое значение. С точки зрения теории важно являются ли они универсальными машинами и можно ли согласно тезису Чёрча-Тьюринга обучить их любому алгоритму. С точки зрения практики важно, какие части трансформера являются ключевыми для полноты, а без каких можно обойтись, как обучать трансформеры алгоритмам, можно ли улучшить архитектуру трансформеров, сделав их более умными.

Трансформеры с непрерывными активациями являются полными по Тьюрингу. К сожалению на данный момент все практические реализации трансформеров не являются полными по Тьюрингу из-за ограниченной точности представления данных. В дальнейшем мы проанализируем математические концепции приведшие к этому выводу и рассмотрим модификации трансформеров, которые позволяют им преодолевать эти ограничения.

2 Машина Тьюринга

Машина Тьюринга — это абстрактная универсальная вычислительная модель, предложенная Аланом Тьюрингом, которая формально описывает алгоритмические вычисления.

2.1 Формальное определение

Машина Тьюринга (Hopcroft, 2001) — это семерка

$$(Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F),$$

состоящая из следующих компонент:

- **Множество состояний** Q: конечное множество состояний, в которых может находиться машина. Содержит начальное состояние $q_0 \in Q$ и, возможно, одно или несколько конечных состояний $F \subseteq Q$.
- **Алфавит ленты** Γ: конечное множество символов, которые могут записываться на ленте. Включает специальный символ пустого места $\sqcup \in \Gamma$.
- **Входной алфавит** Σ: подмножество алфавита ленты ($\Sigma \subseteq \Gamma$), которое используется для формирования входных данных. Входной алфавит не включает символ пустого места $\sqcup$.
- **Лента**: бесконечная в обе стороны последовательность ячеек, каждая из которых может содержать символ из алфавита ленты Γ.
- **Головка чтения/записи**: устройство, которое перемещается вдоль ленты, может читать символы и записывать новые символы.

- **Функция перехода** δ: отображение, определяющее переходы машины:

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\},$$

где L означает перемещение головки влево, а R — перемещение вправо. Функция перехода указывает новое состояние, символ для записи и направление движения головки в зависимости от текущего состояния и символа под головкой.

- **Начальное состояние** $q_0 \in Q$: состояние, в котором машина начинает свою работу.
- **Множество конечных состояний** $F \subseteq Q$: состояния, при достижении которых машина останавливает свою работу.

Обратим внимание, что все операции локальны в смысле передвижения головки и имеют одинаковую сложность, поскольку являются подстановками букв конечного алфавита.

2.2 Эквивалентные модели

Существуют абстрактные вычислительные модели эквивалентные машине Тьюринга, в том числе:

- недетерминированные машины Тьюринга (Hopcroft, 2001);
- лямбда исчисление (Bernays, 1936);
- взаимодействующие последовательные процессы (Хоаровские процессы) (Hoare и др., 1985);
- модель экторов с ограниченным недетерминизмом (Hewitt и др., 1973);
- RAM машина (Cook и Reckhow, 1972);
- магазинный автомат с 2 магазинами (Hopcroft, 2001);
- считающая машина с 2 счетчиками (Hopcroft, 2001);
- неограниченные (тип 0) грамматики Хомского (Chomsky, 1956);
- квантовые вычисления (Fortnow, 2003);

Недетерминированность является аналогом параллельных вычислений для машины Тьюринга.

Особый интерес представляют мультиагентные системы, поскольку оказывается что каждый агент не должен быть полным по Тьюрингу для возникновения более сложного полного по Тьюрингу поведения, если агенты могут создавать новых агентов.

Известен тезис Чёрча-Тьюринга, что машина Тьюринга является самой мощной физически реализуемой, и любой вычислимый алгоритм можно на ней запустить. Этот тезис не является строгим математическим утверждением из-за отсутствия формального определения алгоритма. Однако, если мы считаем наш мир квантовым, то все физически реализуемые машины построены поверх квантовых вычислений. Поскольку квантовые вычисления эквиваленты машине Тьюринга, то тезис Чёрча-Тьюринга становится теоремой без необходимости формализации алгоритмов, так как симуляция другой машины не может работать быстрее, чем машина на которой она запущена.

3 Классы языков

Хомский предложил классификацию языков на основе подстановочных грамматик, которые их разбирают. Подстановочные грамматики состоят из правил в левой части которых стоит шаблон, а в правой подстановка. Наибольший тип 0 является полным по Тьюрингу и представлял наименьший интерес для Хомского ввиду вычислительной сложности.

Поскольку на естественном языке можно выразить любой алгоритм и любые промежуточные данные при его выполнении, то любая универсальная система, моделирующая рассуждения на естественном языке должна быть полна по Тьюрингу.

Грамматики типа 0 эквивалентны нормальным алгоритмам Маркова. Также существуют более современные формализмы не тонущие в "Тьюринговской трясине"такие как монадические трансформеры, которые используя методы функционального программирования позволяют определить шаблон и подстановку через произвольные частично-рекурсивные функции при этом сохраняя в программе структуру грамматики.

Левые части правил подстановки в грамматиках Хомского могут рассматриваться как кодер, а правые как декодер. Таким образом машина, реализующая грамматику может рассматриваться как рекуррентный кодер-декодер. На практике при реализации такого подхода к разбору естественных языков возникают следующие проблемы:

- медленное последовательное применение правил из-за рекурсии;
- недетерминизм;
- отсутствие способа извлечения грамматики в явном виде из корпуса текстов;

Последняя проблема может быть решена путем замены задачи разбора на задачу предсказания слова по контексту. Например при переводе задача ставится не как трансформация с одного языка на другой, а как дописывание текста перевода после текста на исходном языке, где исходный текст рассматривается как префикс перевода.

4 Проблема остановки

Почему полнота по Тьюрингу имеет практическое значение. В качестве забавного факта в курсе дискретной математики приводится доказательство алгоритмической неразрешимости проблемы остановки. Также в современной математической логике принято сводить доказательство теоремы Гёделя о неполноте к проблеме остановки. К сожалению с точки зрения практики проблема остановки является не более чем курьезом, поскольку любые реальные машины имеют ограниченную память.

Теорема 1 *На машине с конечной памятью проблема остановки разрешима за конечное время.*

Доказательство. Пусть не умаляя общности машина имеет N однобитных ячеек памяти. Рассмотрим все возможные состояния памяти, которых 2^N. Тогда по принципу Дирихле машина за 2^N шагов либо остановится либо перейдет в уже посещенное состояние, то есть зациклится. Таким образом проблема остановки гарантированно разрешима не более чем за 2^N шагов. □

Давайте сформулируем какое-то более практичное утверждение.

Теорема 2 *Задача остановки машины Тьюринга за T шагов разрешима на машине Тьюринга не быстрее чем за T шагов.*

Доказательство. Докажем методом диагонализации. Пусть $h_T(\#i, \#x)$ программа проверяющая что программа с номером $\#i$ на данных $\#x$ завершается за время $t \leq T$ на машине Тьюринга. Докажем, что эта программа не завершается за время $t < T$. Определим $h_T(\#i, \#x)$.

$$h_T(\#i, \#x) = \begin{cases} 1 & \text{if } i(x) \text{ halt in } T \text{ steps,} \\ 0 & \text{otherwise.} \end{cases}$$

Пусть она завершается за время $t \leq T - 1$. Рассмотрим $g_T(\#i)$

$$g_T(\#i) = \begin{cases} 0 & \text{if } h_T(\#i, \#i) = 0, \\ \text{hang} & \text{otherwise.} \end{cases}$$

Тогда верно одно из двух:

1. $h_T(\#g_T, \#g_T) = 0 \to g$ не завершается за время $t + 1 \leq T$ и $g_T(\#g_T) = 0$. Но g_T на шаг длиннее h_T и выдает 0 за время $t + 1 \leq T$, то есть завершается, противоречие.
2. $h_T(\#g_T, \#g_T) = 1 \to g_T(\#g_T)$ не завершается, тогда $h_T(\#g_T, \#g_T) = 0$, противоречие. □

Это утверждение говорит, что нельзя решить проблему остановки за T шагов быстрее чем за T шагов. Оно практически применимо и не является неожиданным, поскольку соответствует интуитивному представлению, что машина Тьюринга – самая быстрая, а значит есть программы, которые нельзя на ней ускорить. Если мы заменим T на бесконечность ω, то используя основную теорему нестандартного анализа получим обычную формулировку утверждения о неразрешимости проблемы остановки.

Следствие 1 *Быстрейший способ проверки завершения машины Тьюринга на входе x за T шагов - запуск на T шагов.*

Доказательство. *По теореме 2 быстрее чем за T шагов. Запуск на T шагов, требует ровно T шагов.* □

Давайте сформулируем и докажем еще более полезное на практике утверждение. Что эту же задачу нельзя разрешить на параллельной машине Тьюринга.

Теорема 3 *Задача остановки машины Тьюринга за T шагов разрешима на недетерминированной машине Тьюринга не быстрее чем за T шагов.*

Доказательство. Чтобы определить, останавливается ли машина Тьюринга на входе x за T шагов, необходимо проверить все возможные состояния её ленты и головки на каждом шаге до T. Даже если мы недетерминированно "угадаем"путь вычислений, нам всё равно нужно подтвердить корректность этого пути, что по лемме 1 требует как минимум T шагов.

Формализуем эти рассуждения. Пусть $h_T(\#i, \#x)$ недетерминированная программа проверяющая что программа с номером $\#i$ на данных $\#x$ завершается за время $t \leq T$ на машине Тьюринга с детерминированным оракулом выбора правил. Если программа с номером $\#i$ детерминированная, то оракул не влияет на работу машины. Докажем, что эта программа не завершается за время $t < T$. Определим недетерминированну программу для всех $j \in [0, 1)$ как $h_t(\#i, \#x, j)$, где j – оракул, определяющий выбор правил, можно считать, что каждому j соответствует детерминированная машина

$$h_T(\#i, \#x, j) = \begin{cases} 1 & \text{if } i(x) \text{ halt in } T \text{ steps,} \\ 0 & \text{otherwise.} \end{cases}$$

Для машины с номером j $h_T(\#i, \#x)$ – детерминированная программа, поскольку используется не более T запросов к оракулу, которые могут быть закодированы как константы в программе. Пусть она завершается для любых $\#i, \#x$ за время $t \leq T-1$ для некоторой машины с номером$j_T(\#i, \#x)$. Поскольку h_T вычислима за T, то функция $j_T(\#i, \#x)$ вычислима и везде определена. Рассмотрим недетерминированную $g_T(\#i)$.

$$g_T(\#i, j) = \begin{cases} 0 & \text{if } h_T(\#i, \#i, j) = 0, \\ \text{hang} & \text{otherwise.} \end{cases}$$

Данное преобразование, не увеличивает количество запросов к оракулу, так что оракул не меняется. Тогда для машины с номером $k = j_T(\#g_T, \#g_T)$ верно одно из двух:

1. $h_T(\#g_T, \#g_T, k) = 0$, тогда $g_T(\#g_T, k) = 0$ и $h_T(\#g_T, \#g_T, k)$ не завершается за время $t \leq T$. Но g_T на шаг длиннее h_T и на машине с номером k выдаёт 0 за время $t + 1 \leq T$, противоречие.

2. $h_T(\#g_T, \#g_T, k) = 1$, тогда $g_T(\#g_T, k)$ не завершается, и $h_T(\#g_T, \#g_T, k) = 0$, противоречие. □

Смысл этого утверждения, что есть задачи которые не ускоряются от параллелизма что уже не так интуитивно ожидаемо.

5 Признаки машины Тьюринга

Давайте сформулируем простые и легко проверяемые признаки машины Тьюринга:

1. Может долго считать;

2. Не забывает что уже посчитала.

Формализуем долгий счет, что сравнительно просто.

Теорема 4 *Время работы не зависит от размера входных данных.*

Доказательство. Предположим обратное. Тогда все программы останавливаются. Тогда проблема остановки разрешима за 1 шаг. □

Формализуем забывание. При забывании данные либо явно затираются либо перестают быть доступными для дальнейших вычислений.

Теорема 5 *Для любой программы P существует эквивалентная программа P', такая что для любого T любые входные и промежуточные данные вычисленные до момента T могут быть доступны машине в момент времени $t > T$.*

Сначала докажем вспомогательное утверждение.

Лемма 1 *Машина Тьюринга с перезаписью ячеек ленты эквивалентна машине с однократной перезаписью.*

Доказательство. Будем моделировать каждую ячейку исходной машины 4 ячейками новой ленты (основная, запасная, метка копирования, метка положения головки). При перезаписи ячейки будем записывать новое значение в запасную ячейку. При симуляции исходной шага исходной машины:

1. последовательно копируются все заполненные ячейки ленты на свободное место справа, скопированные ячейки помечаются;
2. головка позиционируется используя метку положения;
3. выполняется шаг исходной машины;
4. помечается текущее положение головки.

Поскольку головка на каждом шаге всегда сдвигается, а для перезаписи используется запасная ячейка, то никакие метки и символы не перезаписываются. □

Доказательство. По лемме любая абстрактная машина эквивалентна машине Тьюринга с однократной записью. Следовательно она может не затирать данные. Построим следующую цепочку трансформаций

$$M \to T \to W1 \to M$$

Удалим у машины Тьюринга T все конечные состояния и зациклим их. Такая трансформация машин индуцирует эквивалентное преобразование программ $P \to P'$ для машины M. Поскольку при эмуляции шага машины Тьюринга машина с однократной записью копирует всю ленту, то найдется такой $t > T$ для любого символа записанного на ленте в момент времени T, что машина M получит к нему доступ при копировании.□

В отличие от эквивалентности, которая требует сложных доказательств, эти признаки могут быть легко проверены и без длинных математических рассуждений.

Также можно определить и количественный признак машины Тьюринга ограниченной по времени работы T. Количество ячеек памяти такой машины должно быть не меньше чем максимальное количество операций записи в память за это время.

$$M_T \geq W(T).$$

Но поскольку W – очевидно монотонна, то зная M мы можем определить максимальное время работы машины.

$$T \leq W^{-1}(M).$$

6 Трансформеры

До трансформеров основной архитектурой для работы с текстами были рекуррентные сети (RNN) (Sutskever и др., 2014). Основным недостатком их были медленные последовательные вычисления из-за зависимости от состояния.

Сформулируем практические требования для эффективной нейросетевой системы рассуждений на естественном языке.

- простота обучения: обучение с учителем на корпусе текстов
- выигрыш в скорости от параллельных вычислений

- однородность вычислений: фиксированное количество вычислений за один шаг

Фактически здесь ставится задача векторизации вычислений. Стандартный подход для решения этой задачи используется в параллельном и функциональном программировании. Зависимость между данными выражается явным образом в виде чистой функции. Поскольку внутри нее нет зависимости от состояния, все независимые пути вычислений могут быть выполнены параллельно. Эти требования были реализованы в архитектуре трансформеров. По сравнению с предыдущими архитектурами это позволило получить как выигрыш в скорости обучения и вывода, так и в качестве работы при том же количестве параметров модели. Что закономерно в дальнейшем привело к взрывному росту размера и качества моделей.

Трансформер (Vaswani и др., 2017) является реализацией модели предсказания последовательностей символов (sequence transducer) в виде чистой функции с внешней итерацией, то есть монадой в терминах функционального программирования. В отличие от RNN архитектура одного шага вычислений является ациклической. Однако если включить итерацию в сам трансформер как обратную связь, то он становится частным случаем RNN.

6.1 Основные компоненты

6.1.1 Механизм самовнимания

Механизм самовнимания позволяет модели оценивать важность различных частей входной последовательности при генерации выходного представления. Он основан на вычислении внимания по следующим формулам:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

где:

- Q — матрица запросов (queries).
- K — матрица ключей (keys).
- V — матрица значений (values).
- d_k — размерность ключей.

Сходство с запросом Q к базе данных $\{(K, V)\}$ не случайно. По замечанию Алекса Грейвса (Graves, 2020) механизм внимания это то, что остается от памяти если убрать время. Таким образом, механизм внимания является естественным результатом размотки итераций циклов в RNN.

2. Многоголовое внимание

Многоголовое внимание позволяет модели изучать различные виды отношений между словами в предложении. Каждая “голова” внимания вычисляется независимо, а затем их результаты объединяются:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O,$$

где каждая голова определяется как:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),$$

а W_i^Q, W_i^K, W_i^V — матрицы весов для запросов, ключей и значений соответственно.

6.1.2 Позиционное кодирование

Позиционное кодирование является еще одной разновидностью механизма внимания. Поскольку архитектура Transformer не использует рекуррентные связи, т.е. память, для учета порядка слов в последовательности, то позиционное кодирование добавляется к входным данным. Для позиционного кодирования используются гармонические функции:

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right),$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right),$$

где pos — позиция токена, а i — индекс размерности.

Гармоническое позиционное кодирование решает и еще одну задачу – управление забыванием. С точки зрения устойчивости контекст вычислений на одном шаге должен быть ограничен. Стандартным решением проблемы ограничения контекста в RNN является экспоненциальное забывание по времени. Но оно приводит к потере важной контекстной информации, находящейся далеко от текущего положения в последовательности. Эта проблема не является специфичной для трансформеров и рассматривалась в более ранних публикациях, например (Gers и др., 2000).

Гармоническое позиционное кодирование позволяет реализовать внимание в частотной области, где расстояние не зависит от времени. Этот подход может быть перенесен в RNN, например для синтеза музыки (Huang и др., 2018).

6.1.3 Полносвязные нейронные сети

Каждый слой кодера и декодера включает в себя полносвязную нейронную сеть, которая применяется к каждому элементу входной последовательности независимо:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2,$$

где W_1 и W_2 — матрицы весов, а b_1 и b_2 — смещения.

6.1.4 Нормализация слоя и остаточные связи

- **Нормализация слоя** (Ba и др., 2016): применяется для стабилизации и ускорения процесса обучения путем нормализации выходных значений слоя. В отличие от сверточных сетей CNN, где применяется нормализация по пактам при обучении, нормализация по слою применяется как при обучении так и при выводе. Нормализация по слою не зависит от размера пакета, что позволяет применить ее к последовательностям переменной длины.

- **Остаточные связи** (He и др., 2016): добавляют вход слоя к его выходу, что помогает сохранить информацию и улучшает распространение градиента.

Эти особенности являются неотделимой частью современных нейросетей и обеспечивают устойчивость метода обратного распространения ошибки при обучении.

6.2 Структура кодера и декодера

6.2.1 Кодер

Кодер состоит из N идентичных слоев, каждый из которых включает в себя:

1. Многоголовое самовнимание.
2. Полносвязную нейронную сеть.

На выходе каждого слоя используется нормализация слоя и остаточная связь.

6.2.2 Декодер

Декодер также состоит из N идентичных слоев, но включает дополнительный слой внимания, который обращается к выходу кодера. Каждый слой содержит:

1. Маскированное многоголовое самовнимание.
2. Многоголовое внимание к выходу кодера.
3. Полносвязную нейронную сеть.

7 Полнота рекуррентных нейросетей и трансформеров

Рекуррентные нейросети на первый взгляд не обладают бесконечным количеством ячеек памяти, что в соответствии вторым признаком говорит об их неполноте. Однако это не так. По аналогии со считающей машиной с 2 счетчиками достаточно конечного количества ячеек но с бесконечной точностью представления чисел. Таким образом рекуррентная нейросеть с непрерывными активациями является полной по Тьюрингу (Siegelmann и Sontag, 1992). Но любая практическая реализация без внешней памяти не будет обладать этим свойством.

Тот же вывод можно распространить и на архитектуру трансформеров (Pérez и др., 2021). Для трансформеров ключевым оказывается позиционное кодирование в механизме внимания. При использовании гармонического кодирования можно явно указать на математические факты, ограничивающие возможности обращения к памяти для численных реализаций трансформеров. Это теорема Шеннона-Хартли и теорема Котельникова (Shannon, 1948), которые дают верхний предел количества информации при численном гармоническом анализе.

Однако трансформеры, имеют и свои специфические ограничения, связанные с первым признаком неполноты. Они слишком мало думают над ответом. Целевая функция трансформеров не оценивает качество ответа на поставленный вопрос, а только его вероятность в распределении обучающей выборки. Поскольку большая часть корпуса текстов – это беллетристика, то сложно ожидать там примеров качественных логических рассуждений. Таким образом ответы трансформера обычно содержат много воды и самоповторений и мало конкретики. Трансформер может пропускать важные шаги в алгоритмах, поскольку алгоритм с пропущенным шагом почти также правдоподобен. Поскольку у трансформера нет операции удаления символов, то он не может удалять в процессе вывода ошибочные или бесполезные суждения.

8 Модификации трансформеров

Как видно из предыдущего на данный момент трансформеры не удовлетворяют ни первому ни второму признакам машины Тьюринга. Они слишком мало думают, и быстро забывают. На данный момент основным направлением улучшения является именно увеличение времени размышлений. Для новой модели OpenAI O1 зависимость качества ответа от времени вычислений подтверждается экспериментально (OpenAI, 2024).

Существуют несколько работающих подходов, позволяющих улучшить работу трансформеров, сделав их более похожими на машину Тьюринга по этому признаку.

- Рассуждение по шагам. Было замечено, что если добавить в запрос фразу "по шагам-(Wei и др., 2022), то трансформеры резко умнеют. Это можно объяснить тем, что они начинают подражать не литературному тексту из обучающей выборки, а логическим рассуждениям. Существует боле строгое математическое рассуждение, почему так происходит. Без "по шагам"сеть пытается сразу дать ответ. Но информация об ответе в извлекаемом виде не содержится ни в контексте ни в весах самой сети, таким образом сеть дает ответ наугад и часто ошибается.

- Автор-критик. Также было замечено, что диалоговые системы достигают результата за несколько итераций. При этом пользователь выступает в роли критика, отвергая несостоятельные гипотезы трансформера и обеспечивает дополнительный цикл внешней итерации, позволяя трансформеру работать дольше. Но в роли критика не обязательно должен выступать пользователь. Подход автор-критик используется, например, в архитектуре нейросетей GAN (Goodfellow и др., 2014) и позволяет ампутировать лишние пальцы и прочие части тела у творений нейро-художников. Судя по описаниям именно этот подход используется в сети OpenAI O1.

- Мультиагентные системы. Система автор-критик содержит 2 независимых агентов. Полнота по Тьюригу модели экторов и CSP говорит об увеличении мощности системы с ростом числа агентов. Один из возможных подходов – моделирование ролей, исполняемых биологическими агентами, например, архитектор, разработчик, тестировщик, дизайнер, с использованием Microsoft Autogen (Wu и др., 2023) или аналогичных.

- Вычисления с оракулом. В мультиагентной системе не все агенты должны быть трансформерами. Могут быть агенты поиска в интернет, поиска в локальной базе данных, агент запускающий произвольные программы, например тесты. Такие агенты могут рассматриваться как оракулы (Turing, 1939). При этом они сами могут быть полными по

Тьюрингу, например универсальная вычислительная машина. Тогда трансформер выполняет только роль транслятора с подмножества человеческого языка на машинный язык, и для полноты по Тьюрингу всей системы, его полнота не требуется.

9 Заключение

По признакам машины Тьюринга сферический естественный интеллект в вакууме не является полным по Тьюрингу, поскольку постоянно что-то забывает. Поэтому трансформеры могут его превзойти и без изменения архитектуры с использованием цепочки рассуждений. Чтобы сделать полным по Тьюрингу интеллект надо снабдить бесконечным блокнотом. Куда вставить бесконечный блокнот на данный момент является неразрешенной инженерной проблемой в области развития больших языковых моделей на основе нейросетей.

Еще одной проблемой является проблема оплаты вычислений. Поскольку для обычных трансформереров количество сгенерированных слов зависит от размера ввода, то пользователь может оплачивать их работу пословно. Модифицированный трансформер, удовлетворяющий признакам машины Тьюринга может работать сколь угодно долго не выдавая ответа. При этом пользователь должен будет оплачивать время вычислений, не контролируя, что именно вычисляется. Здесь прослеживается прямая аналогия с наукометрическими показателями и финансированием научных исследований.

Список литературы

Ba, Jimmy Lei, Kiros, Jamie Ryan и Hinton, Geoffrey E. (2016). «Layer Normalization», *arXiv preprint arXiv:1607.06450*,

Bernays, Paul (1936). «Alonzo Church. An unsolvable problem of elementary number theory. American journal of mathematics, vol. 58 (1936), pp. 345–363.» *The Journal of Symbolic Logic*, Vol. 1 No. 2, с. 73—74.

Chomsky, Noam (1956). «Three models for the description of language», *IRE Transactions on information theory*, Vol. 2 No. 3, с. 113—124.

Cook, Stephen A и Reckhow, Robert A (1972). «Time-bounded random access machines», *Proceedings of the fourth annual ACM symposium on Theory of computing*, с. 73—80.

Fortnow, Lance (2003). «One complexity theorist's view of quantum computing», *Theoretical Computer Science*, Vol. 292 No. 3, с. 597—610.

Gers, Felix A, Schmidhuber, Jürgen и Cummins, Fred (2000). «Learning to forget: Continual prediction with LSTM», *Neural computation*, Vol. 12 No. 10, с. 2451—2471.

Goodfellow, Ian и др. (2014). «Generative adversarial nets», *Advances in neural information processing systems*, Vol. 27.

Graves, Alex (2020). *Attention and Memory in Deep Learning*, Video lecture, DeepMind and University College London (UCL). Available at: `https://www.youtube.com/watch?v=AIiwuClvH6k&ab_channel=DeepMind`. DeepMind.

He, Kaiming и др. (2016). «Deep residual learning for image recognition», *Proceedings of the IEEE conference on computer vision and pattern recognition*, с. 770—778.

Hewitt, Carl, Bishop, Peter и Steiger, Richard (1973). «Session 8 formalisms for artificial intelligence a universal modular actor formalism for artificial intelligence», *Advance papers of the conference*. Т. 3. Stanford Research Institute Menlo Park, CA, с. 235.

Hoare, Charles Antony Richard и др. (1985). *Communicating sequential processes*, т. 178. Prentice-hall Englewood Cliffs.

Hopcroft, JE (2001). *Introduction to Automata Theory, Languages, and Computation*,

Huang, Cheng-Zhi Anna и др. (2018). «Music transformer», *arXiv preprint arXiv:1809.04281*,

OpenAI (2024). *Learning to Reason with LLMs*, Accessed: 2024-10-28. **available at**: `https://openai.com/index/learning-to-reason-with-llms/`.

Pérez, Jorge, Barceló, Pablo и Marinkovic, Javier (2021). «Attention is turing-complete», *Journal of Machine Learning Research*, Vol. 22 No. 75, с. 1—35.

Shannon, Claude Elwood (1948). «A mathematical theory of communication», *The Bell system technical journal*, Vol. 27 No. 3, с. 379—423.

Siegelmann, Hava T и Sontag, Eduardo D (1992). «On the computational power of neural nets», *Proceedings of the fifth annual workshop on Computational learning theory*, с. 440—449.

Sutskever, Ilya, Vinyals, Oriol и Le, Quoc V. (2014). «Sequence to Sequence Learning with Neural Networks», *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*. MIT Press, с. 3104—3112.

Turing, Alan Mathison (1939). «Systems of logic based on ordinals», *Proceedings of the London Mathematical Society, Series 2*, Vol. 45, с. 161—228.

Vaswani, Ashish и др. (2017). «Attention Is All You Need.(Nips), 2017», *arXiv preprint arXiv:1706.03762*, Vol. 10, S0140525X16001837.

Wei, Jason и др. (2022). «Chain-of-thought prompting elicits reasoning in large language models», *Advances in neural information processing systems*, Vol. 35, с. 24824—24837.

Wu, Qingyun и др. (2023). «Autogen: Enabling next-gen llm applications via multi-agent conversation framework», *arXiv preprint arXiv:2308.08155*,