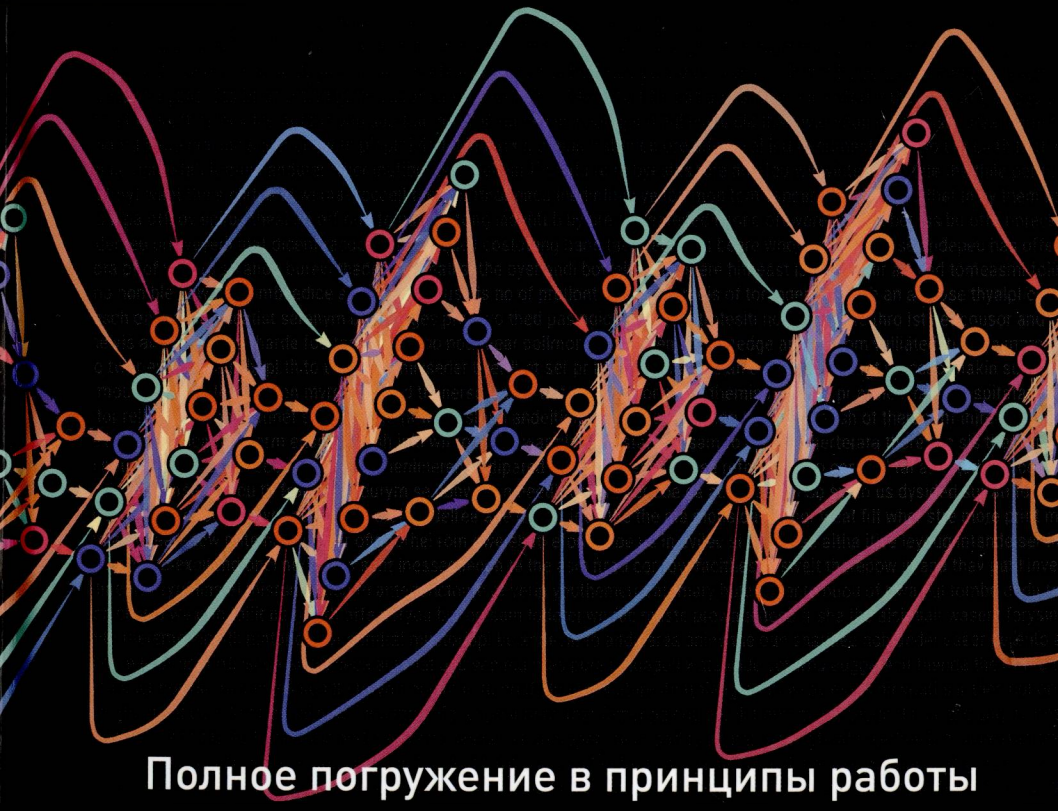
СТИВЕН ВОЛЬФРАМ
Всемирно известный физик-математик
и разработчик WolframAlpha

КАК УСТРОЕН ChatGPT?



Полное погружение в принципы работы
и спектр возможностей
самой известной нейросети в мире

перевод Елены Быковой

МИФ

Эта книга принадлежит

Контакты владельца

Эту книгу хорошо дополняют

От «Энигмы» до ChatGPT
Рустам Агамалиев

ИИ-2041. Десять образов нашего будущего
Кай-Фу Ли, Чэнь Цюфань

От предвидения к власти
Аджай Агравал, Джошуа Ганс, Ави Голдфарб

STEPHEN WOLFRAM

What Is ChatGPT Doing... and Why Does It Work?

Цифровые технологии

СТИВЕН ВОЛЬФРАМ

Как устроен ChatGPT?

ПОЛНОЕ ПОГРУЖЕНИЕ В ПРИНЦИПЫ РАБОТЫ И СПЕКТР ВОЗМОЖНОСТЕЙ САМОЙ ИЗВЕСТНОЙ НЕЙРОСЕТИ В МИРЕ

Перевод с английского
Елены Быковой

Москва
МИФ
2024

УДК 004.032.26
ББК 16.632
В72

Научный редактор Здоров Антон

На русском языке публикуется впервые

Вольфрам, Стивен

В72 Как устроен ChatGPT? Полное погружение в принципы работы и спектр возможностей самой известной нейросети в мире / Стивен Вольфрам ; пер. с англ. Е. Быковой ; науч. ред. А. Здоров. — Москва : МИФ, 2024. — 192 с. — (Цифровые технологии).

ISBN 978-5-00214-604-8

ChatGPT в рекордно короткие сроки завоевал популярность благодаря способности писать тексты на любые темы — почти как человек. Но как именно он это делает? Может ли он понимать, что пишет, и что происходит внутри его искусственного разума?

Стивен Вольфрам — удостоенный наград ученый и создатель программных систем, таких как Wolfram|Alpha, — предлагает понятное и увлекательное объяснение внутреннего устройства ChatGPT.

Узнайте, как ChatGPT объединяет новейшие технологии нейронных сетей с фундаментальными идеями о языке и человеческом мышлении.

УДК 004.032.26
ББК 16.632

Все права защищены.
Никакая часть данной книги
не может быть воспроизведена
в какой бы то ни было форме
без письменного разрешения
владельцев авторских прав.

ISBN 978-5-00214-604-8

What Is ChatGPT Doing ... and Why Does It Work?
© 2023 Stephen Wolfram.
Original English language edition published by Wolfram Media 100 Trade Center Dr. 6th Floor, Champaign Illinois 61820, USA.
Arranged via Licensor's Agent: DropCap Inc. and Igor Korzhenevskiy of Alexander Korzhenevski Agency (Russia). All rights reserved
© Издание на русском языке, перевод, оформление. ООО «Манн, Иванов и Фербер», 2024

ОГЛАВЛЕНИЕ

Предисловие 9

ЧТО ДЕЛАЕТ CHATGPT И ПОЧЕМУ ЭТО РАБОТАЕТ?

Он просто добавляет по одному слову за раз 15

Откуда берутся эти вероятности? 26

Что такое модель? 34

Модели для человеческих задач 38

Нейронные сети 42

Машинное обучение нейронных сетей 59

Обучение нейронных сетей: практика и знания 69

Достаточно большая сеть может сделать все что угодно — так ведь? 82

Концепция эмбеддингов 89

Внутри ChatGPT 99

Обучение ChatGPT 114

Помимо базового обучения 119

Что позволяет ChatGPT работать? ... 123

Смысловое пространство
и семантические законы движения ... 133

Семантика и сила вычислительного языка ... 141

Так что же делает ChatGPT и почему это работает? ... 149

Благодарности ... 153

WOLFRAM|ALPHA КАК СПОСОБ ДОБАВИТЬ В CHATGPT СВЕРХВОЗМОЖНОСТИ ВЫЧИСЛЯЕМЫХ ЗНАНИЙ

ChatGPT и Wolfram|Alpha ... 158

Базовый пример ... 161

И еще несколько примеров ... 166

Перспективы ... 184

Дополнительные ресурсы ... 191

ПРЕДИСЛОВИЕ

Эта небольшая книга представляет собой попытку объяснить, как работает ChatGPT. В некотором смысле это история о технологиях. Но еще и разговор о науке и философии. И для того, чтобы рассказать эту историю, нам придется собрать воедино огромный спектр идей и открытий, сделанных на протяжении многих столетий.

Я сам с большим интересом и волнением наблюдаю, как вещи, которые так долго меня интересовали, объединяются в результате такого прогресса. Сложное поведение простых программ, погружение в суть языка и смыслообразования, а также практические возможности больших компьютерных систем — все это является частью истории ChatGPT.

ChatGPT основан на концепции нейронных сетей, составленной в 1940-х годах в качестве модели работы головного мозга. Я сам впервые спрограммировал нейронную сеть в 1983 году, но тогда ничего интересного из нее не вышло. Однако 40 лет спустя, когда у нас есть компьютеры, считающие в миллион раз быстрее, миллиарды страниц текста в интернете и целый ряд инженерных инноваций, ситуация совершенно иная. И, ко всеобщему удивлению, современная нейронная сеть, которая в миллиард раз больше созданной мной в 1983 году, способна делать то, что раньше считалось под силу только человеку, — генерировать осмысленный текст.

Эта книга, написанная вскоре после дебюта ChatGPT, состоит из двух частей. В первой объясняется, что такое ChatGPT и как ему удается выполнять сугубо человеческую работу по генерированию текста. Вторая посвящена вычислительным инструментам ChatGPT (выходящим за рамки человеческих возможностей) и сверхспособностям нашей системы Wolfram|Alpha в области вычисляемых знаний.

На момент написания книги прошло всего три месяца с запуска ChatGPT, и мы только начинаем понимать последствия этого события — как практические, так

и интеллектуальные. Но пока ChatGPT служит напоминанием о том, что, несмотря на множество уже сделанных изобретений и открытий, сюрпризы всё еще возможны.

Стивен Вольфрам,
28 февраля 2023 года

ЧТО ДЕЛАЕТ CHATGPT И ПОЧЕМУ ЭТО РАБОТАЕТ?

ОН ПРОСТО ДОБАВЛЯЕТ ПО ОДНОМУ СЛОВУ ЗА РАЗ

Тот факт, что ChatGPT может автоматически генерировать текст, который выглядит так, словно написан человеком, поражает наше воображение. Большинство людей недоумевают, как чат-бот выполняет интеллектуальную работу, которая прежде считалась доступной исключительно человеку. Как он это делает? Моя цель — дать вам хотя бы общее представление о том, что происходит внутри ChatGPT, а затем исследовать, как ему удается так хорошо создавать то, что выглядит как осмысленный текст. Хочу сразу сказать, что я собираюсь сосредоточиться на общих принципах работы ChatGPT и, хотя буду упоминать некоторые технические детали, не стану в них углубляться. (Все, о чем я буду говорить, применимо и к другим большим языковым моделям, подобным ChatGPT.)

Первое, что нужно объяснить, — ChatGPT всегда пытается создать «разумное продолжение» любого текста, который у него есть на данный момент. Под словами «разумное продолжение» мы подразумеваем «то, что можно ожидать, исходя из того, что люди уже написали на миллиардах веб-страниц».

Итак, давайте предположим, что у нас есть предложение *«Самое лучшее в ИИ — это его способность...»*. Представьте, что вы прочитываете миллиарды страниц текста, написанных людьми (скажем, веб-страницы в интернете и оцифрованные книги), и находите все аналогичные предложения, а затем смотрите, какое слово чаще всего встречается в них на месте многоточия. ChatGPT делает что-то подобное, за исключением того, что (как я объясню позже) он не просматривает тексты буквально — он ищет фрагменты, которые «совпадают по смыслу». Результатом его работы является список слов, которые логически могут следовать за основным словом, ранжированных по степени вероятности их присутствия в этом предложении:

Самое лучшее в ИИ — это его способность... (*англ.* The best thing about AI is its ability to...)	...учиться (*англ.* learn)	4,5%
	...предсказывать (*англ.* predict)	3,5%
	...создавать (*англ.* make)	3,2%
	...понимать (*англ.* understand)	3,1%
	...делать (*англ.* do)	2,9%

Примечательно, что, когда ChatGPT пытается написать эссе, по сути он просто снова и снова спрашивает: «Учитывая текст на данный момент, каким должно быть следующее слово?» — и каждый раз добавляет это новое слово. (Точнее, он добавляет так называемый токен, который может быть и частью слова; именно поэтому ChatGPT иногда может «составлять новые слова». Но об этом позже.)

Итак, на каждом этапе ChatGPT создает список слов с указанием вероятности их присутствия на данном месте. Но какое из них нужно выбрать, например, для эссе? Можно предположить, что это должно быть слово с самым высоким рейтингом (то есть то, для которого определена самая высокая вероятность). Однако именно здесь ChatGPT начинает проявлять свои вуду-способности. Потому что по какой-то причине (возможно, однажды мы ее даже узнаем), если всегда будем выбирать слово с самым высоким рейтингом, мы получим очень гладенькое эссе без малейшего признака креативности (и которое будет слово в слово повторять множество других текстов). Но если мы наугад выбираем слова с более низким рейтингом, то эссе получается «более интересное».

Здесь действует фактор случайности, а это значит, что, даже используя каждый раз один и тот же

промпт*, мы, скорее всего, получим разные эссе. И, как мы уже упоминали, у ChatGPT есть своя вуду-идея, то есть у него существует так называемый температурный параметр, который определяет, как будут использоваться слова с более низким рейтингом, и для создания эссе, оказывается, лучше всего подходит температура, равная 0,8. (Нужно подчеркнуть, что это никакая не теория — это просто вывод из того, что работает на практике. И например, понятие «температура» применяется потому, что здесь используются экспоненциальные распределения, знакомые нам из статистической физики, но нет никакой физической связи — по крайней мере насколько нам известно.)

Прежде чем мы продолжим, я должен объяснить, что ради простоты изложения не стану рассматривать всю систему, лежащую в основе ChatGPT. Вместо этого буду работать с более простой системой — GPT-2, которую можно запустить на обычном персональном компьютере. И поэтому практически для всего, что вам здесь покажу, я включу код на языке Wolfram, который вы сможете немедленно запустить на своем компьютере.

* Промпт (от *англ.* prompt — «побуждать») — запрос, подсказка или инструкция, которые вы даете искусственному интеллекту. *Прим. пер.*

Например, вот как получить таблицу вероятностей, приведенную выше. Во-первых, мы должны иметь базовую нейронную сеть языковой модели:

```
In[ ]:= model = NetModel[{"GPT-2 Transformer Trained
on WebText Data", "Task" → "LanguageModeling"}]
```



Позже мы заглянем внутрь этой нейронной сети и поговорим о том, как она работает. Но пока можем просто применить эту модель к нашему тексту в качестве черного ящика и запросить пять наиболее подходящих слов, которые, по мнению модели, должны следовать за указанным текстом:

```
In[ ]:= model["The best thing about AI is its ability to",
{"TopProbabilities", 5}]
Out[ ]= {do → 0,0288508, understand → 0,0307805,
make → 0,0319072, predict → 0,0349748, learn → 0,0445305}
```

Затем эти строки превращаются в отформатированный набор данных:

```
In[ ]:= Dataset[ReverseSort[Association[%]],
ItemDisplayFunction → (PercentForm[#, 2] &)]
```

Out[ ]=

learn	4,5%
predict	3,5%
make	3,2%
understand	3,1%
do	2,9%

Вот что произойдет, если многократно «применять модель» — на каждом шаге добавлять слово, имеющее наибольшую вероятность (в коде это слово decision):

In[ ]:= **NestList[StringJoin[#, model[#, "Decision"]] &, "The best thing about AI is its ability to", 7]**

Out[ ] = {Самое лучшее в ИИ — это его способность,
Самое лучшее в ИИ — это его способность учиться,
Самое лучшее в ИИ — это его способность учиться на,
Самое лучшее в ИИ — это его способность учиться на собственном опыте,
Самое лучшее в ИИ — это его способность учиться на собственном опыте.,
Самое лучшее в ИИ — это его способность учиться на собственном опыте. Это,
Самое лучшее в ИИ — это его способность учиться на собственном опыте. Это не,
Самое лучшее в ИИ — это его способность учиться на собственном опыте. Это не так.}

Что будет, если мы продолжим предложение? В этом случае (при нулевой температуре) полученный текст оказывается запутанным и изобилующим повторами:

> Самое лучшее в ИИ — это его способность учиться на собственном опыте. Это не просто вопрос обучения на своем опыте, это изучение окружающего вас мира. Искусственный интеллект — очень хороший пример. Это очень хороший пример того, как использовать искусственный интеллект для улучшения своей жизни. Это очень хороший пример того, как использовать искусственный интеллект для улучшения своей жизни. Искусственный интеллект — очень хороший пример того, как использовать его для улучшения своей жизни. Это очень хороший пример того, как использовать искусственный интеллект для

Но что, если вместо того, чтобы всегда выбирать самое популярное слово, мы выберем слова с меньшей вероятностью (с температурой 0,8)? Тогда может получиться следующий текст:

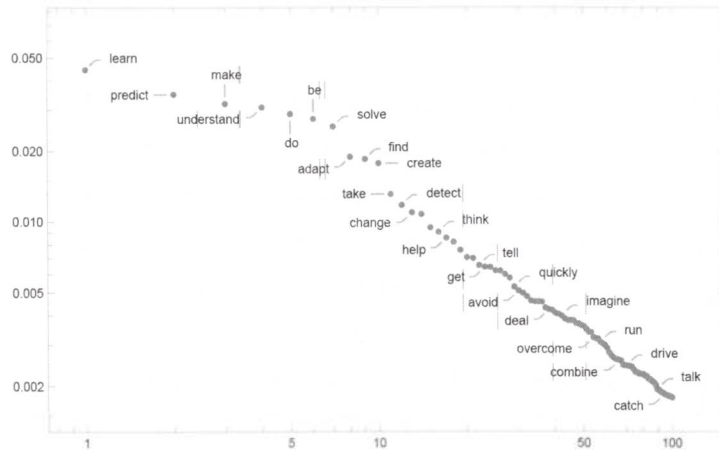
> {Самое лучшее в ИИ — это его способность,
> Самое лучшее в ИИ — это его способность создавать,

Самое лучшее в ИИ — это его способность создавать миры,
Самое лучшее в ИИ — это его способность создавать миры, которые,
Самое лучшее в ИИ — это его способность создавать миры, которые являются,
Самое лучшее в ИИ — это его способность создавать миры, которые являются одновременно захватывающими,
Самое лучшее в ИИ — это его способность создавать миры, которые являются одновременно захватывающими,}

И каждый раз, когда мы будет выбирать случайные слова, текст окажется разным, как в следующих примерах:

Самое лучшее в ИИ — это его способность к обучению. Мне всегда нравилось
Самое лучшее в ИИ — это его способность по-настоящему проникать в ваш мир и просто
Самое лучшее в ИИ — это его способность изучать поведение человека и то, как он
Самое лучшее в ИИ — это его способность проделывать отличную работу по обучению нас
Самое лучшее в ИИ — это его способность создавать реальные задачи, но вы можете

Стоит отметить, что даже на первом этапе имеется множество возможных «следующих слов» (при температуре 0,8), хотя вероятность их использования падает достаточно быстро (данные на этом логарифмическом графике демонстрируют зависимость от n^{-1}, которая очень характерна для общей статистики языка):



Так что же произойдет, если мы продолжим создавать текст? Вот этот случайный пример уже лучше, чем пример с самыми популярными (нулевая температура) словами, но текст все еще немного странный:

> Самое лучшее в ИИ — это его способность видеть насквозь и осмысливать мир вокруг нас, а не паниковать и игнорировать. Мы также можем сказать, что ИИ «выполняет свою работу» или «заурядный ИИ».

> Действительно, делая бесконечное количество шагов, разработка машины, которая может быть интегрирована с другими системами или управляться одной системой, является одним из наиболее фундаментальных процессов ИИ. Помимо взаимодействия человека и машины ИИ также играет важную роль в творчестве.

Этот текст был написан простой моделью GPT-2 (2019 года выпуска). У более современной модели, GPT-3, результаты получаются намного лучше. Следующий текст был создан при помощи самых популярных (нулевая температура) слов с использованием того же самого промпта, но с помощью новой модели — GPT-3:

> Самое лучшее в ИИ — это его способность автоматизировать процессы и принимать решения быстро и точно. ИИ можно использовать для автоматизации повседневных задач вроде ввода данных или для принятия сложных решений, таких как прогнозирование поведения клиентов или анализ больших наборов данных. ИИ также можно использовать для улучшения обслуживания клиентов, поскольку он позволяет быстро и точно реагировать на запросы клиентов. ИИ также можно использовать для повышения точности медицинских диагнозов и автоматизации процесса открытия новых лекарств.

А вот пример, созданный при температуре 0,8:

> Самое лучшее в ИИ — это его способность учиться и развиваться с течением времени, что позволяет ему постоянно улучшать свою производительность и быть более эффективным в выполнении задач. ИИ может использоваться для автоматизации повседневных задач, позволяя людям сосредоточиться на более важных вещах. ИИ также можно использовать для принятия решений и предоставления информации, которую в противном случае людям было бы невозможно понять.

ОТКУДА БЕРУТСЯ ЭТИ ВЕРОЯТНОСТИ?

Итак, ChatGPT всегда выбирает следующее слово, основываясь на вероятностях. Но откуда берутся эти вероятности? Давайте начнем с более простой задачи. Рассмотрим создание текста на английском языке по одной букве (а не по слову) за раз. Как мы можем определить вероятность для каждой буквы?

Самое простое, что мы могли бы сделать, — это взять образец английского текста и подсчитать, как часто в нем встречается каждая буква. Например, так подсчитываются буквы в статье о кошках (*англ.* cats) в «Википедии:

```
In[ ]:= LetterCounts[WikipediaData["cats"]]
Out[ ]= <|e → 4279, a → 3442, t → 3397, i → 2739, s → 2615,
n → 2463, o → 2426, r → 2147, h → 1613, l → 1552, c → 1405,
d → 1331, m → 989, u → 916, f → 760, g → 745, p → 651,
```

```
y → 591, b → 511, w → 509, v → 395, k → 212, T → 114, x → 85,
A → 81, I → 68, S → 55, F → 42, z → 38, F → 36...|>
```

А это в статье о собаках (*англ.* dogs):

```
In[ ]:= LetterCounts[WikipediaData["dogs"]]
Out[ ]= <|e → 3911, a → 2741, o → 2608, i → 2562, t → 2528,
s → 2406, n → 2340, r → 1866, d → 1584, h → 1463,
l → 1355, c → 1083, g → 929, m → 859, u → 782, f → 662,
p → 636, y → 500, b → 462, w → 409, v → 406, k → 151,
T → 90, C → 85, I → 80, A → 74, x → 71, S → 65...|>
```

Результаты похожи, но не идентичны (буква *o*, без сомнения, чаще встречается в статье о собаках, потому что, в конце концов, она присутствует в самом слове dog). Тем не менее, если мы возьмем достаточно большую выборку текстов на английском языке, то можем ожидать, что в итоге получим достаточно схожие результаты:

```
In[ ]:= [ English LANGUAGE [ character frequencies ]
Out[ ]= {e → 12,7%, t → 9,06%, a → 8,17%, o → 7,51%,
i → 6,97%, n → 6,75%, s → 6,33%, h → 6,09%, r → 5,99%,
d → 4,25%, l → 4,03%, c → 2,78%, u → 2,76%, m → 2,41%,
w → 2,36%, f → 2,23%, g → 2,02%, y → 1,97%, p → 1,93%,
b → 1,49%, v → 0,978%, K → 0,772%, j → 0,153%, x → 0,150%,
q → 0,0950%, z → 0,0740%}
```

Вот что мы получим, если просто сгенерируем последовательность букв с такими вероятностями:

> rronoitadatcaeaesaotdoysaroiyiinnbantoioestlhddeocn
> eooewceseciselnodrtrdgriscsatsepesdcniouhoetsedeyh
> edslernevstothindtbmnaohngotannbthrdthtonsipieldn

Мы можем разбить это на «слова», добавив пробелы, как если бы это были буквы с определенной вероятностью:

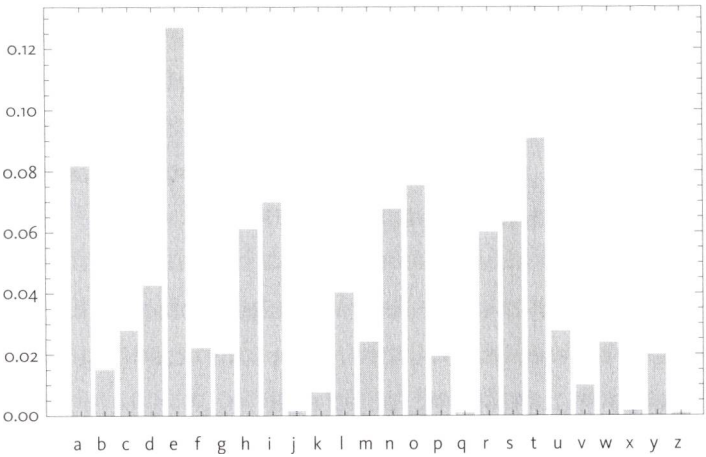
> sd n oeiaim satnwhoo eer rtr ofiianordrenapwokom del
> oaas ill e h f rellptohltvoettseodtrncilntehtotrkthrslo
> hdaol n sriaefr hthehtn ld gpod a h y oi

Мы также можем упростить процесс, разделив буквы в соответствии со стандартной длиной слов в английском языке:

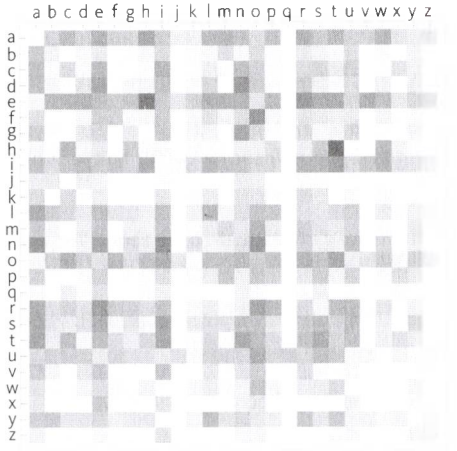
> ni hilwhuei kjtn isjd erogofnr n rwhwfao rcuw lis fahte
> uss cpnc nluoe nusaetat llfo oeme rrhrtn xdses ohm oa
> tne ebedcon oarvthv ist

Нам не удалось обнаружить здесь никаких реально существующих слов, но последующие результаты выглядят немного лучше. Однако, чтобы пойти дальше, нам нужно сделать нечто большее, чем просто выбрать буквы наугад. Так, например, мы знаем, что если у нас есть буква *q*, то следующей буквой должна быть *u*.

Вот график вероятностей самих букв:



А вот график, который показывает вероятности пар букв в типичном английском тексте. Первая буква пары показана сверху, вторая — слева:



Здесь мы видим, например, что столбец *q* пуст (нулевая вероятность), за исключением пересечения со строкой *u*. Итак, теперь вместо того, чтобы генерировать наши «слова» по одной букве за раз, давайте генерировать их, рассматривая по две буквы за раз. Вот результат, который, как оказалось, включает в себя несколько реальных слов:

> on inhuman men ise forernoft wheat iofobato buc is corey ousesetiv fallen tinoco rye for a the ecseri pasuthrgr cuconom tra tesla will tat pere thi

Имея достаточно большой объем текста на английском языке, мы можем определить вероятности не только отдельных букв или их пар, но и более длинных серий (*n*-буквенные). И если мы будем генерировать случайные «слова», постепенно увеличивая это число *n*, то увидим, что получающиеся «слова» становятся более реалистичными:

0 on gxeeetowmt tsifhy ah aufnsoc ior oia itlt bnc tu ih uls

1 ri io os ot timumumoi gymyestit ate bshe abol viowr wotybeat mecho

2 wore hi usinallistin hia ale warou pothe of premetra bect upo pr

3 qual musin was witherins wil por vie surgedygua was suchinguary outheydays theresist

4 stud made yello adenced through theirs from cent intous wherefo proteined screa

5 special average vocab consumer market prepara injury trade consa usually speci utility

Но давайте предположим, что мы имеем дело с целыми словами, а не с отдельными буквами (как и делает ChatGPT). В английском языке насчитывается около 40 тысяч достаточно часто используемых слов. И, просмотрев большой корпус английского текста (скажем, несколько миллионов книг, содержащих в общей сложности несколько сотен миллиардов слов), мы можем определить, насколько часто встречается каждое слово. Используя эту информацию, можно начать генерировать «предложения», в которых каждое слово выбирается независимо и случайным образом на основе той вероятности, которой оно обладает в корпусе. Вот пример того, что мы получаем:

> of program excessive been by was research rate not here of of other is men were against are show they the different the half the the in any were leaved

Безусловно, получается полная ерунда. Итак, как же добиться лучших результатов? Точно так же, как с буквами, мы можем учитывать вероятности не только для отдельных слов, но и для словосочетаний из двух и большего числа слов. Вот пять примеров

того, что можно получить, если начать генерировать предложение, начинающееся со слова «кошка»:

> cat through shipping variety is made the aid emergency can the
>
> cat for the book flip was generally decided to design of
>
> cat at safety to contain the vicinity coupled between electric public
>
> cat throughout in a confirmation procedure and two were difficult music
>
> cat on the theory an already from a representation before a

Здесь текст становится более читабельным. Теперь мы можем представить, что, если бы использовали достаточно длинные словосочетания, мы бы, по сути, получили чат — в том смысле, что это нечто генерировало бы последовательности слов длиной в эссе с «правильными общими вероятностями написания эссе». Но проблема заключается в том, что у нас нет такого количества написанных на английском языке текстов, чтобы можно было вывести эти вероятности.

В интернете, возможно, содержится несколько сотен миллиардов слов, в книгах, которые были оцифрованы, может содержаться еще сто миллиардов. Но при наличии 40 тысяч общеупотребительных слов число возможных словосочетаний из двух слов составляет

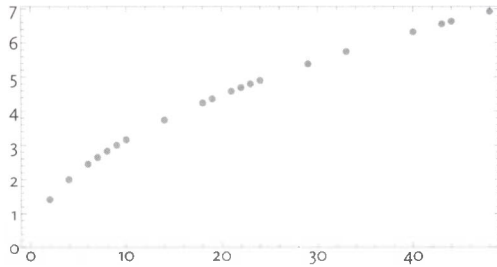
1,6 миллиарда, а число возможных комбинаций из трех слов равно 60 триллионам. Таким образом, мы никак не можем оценить вероятности для всех этих случаев на основе имеющегося текста. И к тому времени, когда мы добираемся до «фрагментов эссе» длиной в 20 слов, количество возможных комбинаций превышает количество частиц во Вселенной, так что все эти вероятности никогда не смогут быть подсчитаны.

Что мы можем сделать? Главная идея заключается в том, чтобы создать модель, которая позволит нам оценить вероятность последовательностей, даже несмотря на то, что мы никогда не видели эти последовательности непосредственно в тексте. Именно такая модель — большая языковая модель (БЯМ, *англ.* large language model, LLM) — лежит в основе ChatGPT, и она была специально создана для того, чтобы выполнять оценку этих вероятностей.

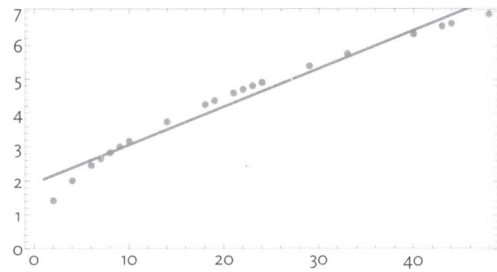
ЧТО ТАКОЕ МОДЕЛЬ?

Предположим, что вы (подобно Галилею в конце 1500-х годов) хотите знать, сколько времени будет падать пушечное ядро, если сбросить его поочередно со всех этажей Пизанской башни. Вы можете просто замерить время в каждом конкретном случае и составить таблицу результатов. Или сделать то, что является сутью теоретической науки: создать модель, которая позволит вам вычислить ответ, вместо того чтобы проводить серию экспериментов и получать данные эмпирическим путем.

Давайте представим, что у нас есть (несколько усредненные) данные о том, сколько времени требуется пушечному ядру, чтобы долететь до земли при падении с разных этажей:



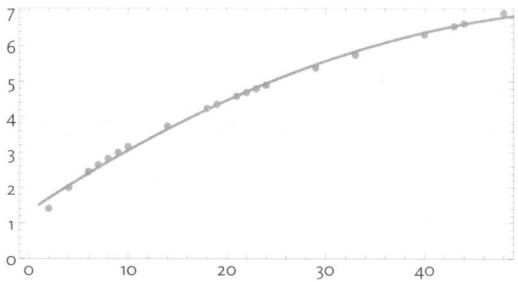
Как мы можем выяснить, сколько времени потребуется ядру, чтобы долететь до земли при падении с конкретного этажа, если не проводили эксперимент? В этом случае мы можем использовать известные законы физики. Но, допустим, все, что у нас есть, — это экспериментальные данные, и мы не знаем, какие основополагающие физические законы управляют движением ядра. Тогда мы могли бы сделать математическое предположение, например, что, возможно, нам следует использовать в качестве модели прямую линию:



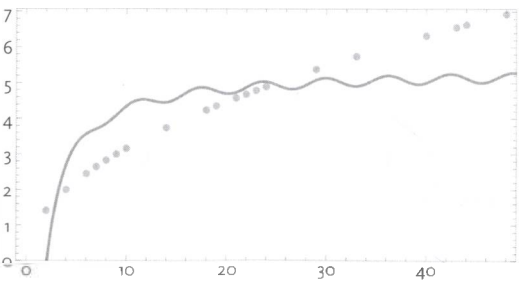
Можно было бы провести разные прямые линии. Но эта линия в среднем ближе всего к полученным нами

данным. И по этой прямой мы можем оценить время падения ядра с любого этажа.

Как мы поняли, что нужно провести прямую линию? Это следует из самых основ математики. Мы привыкли к тому, что множество эмпирически полученных данных хорошо описываются с помощью простых математических формул. Мы могли бы попробовать что-нибудь математически более сложное, скажем: $a + bx + cx^2$, и тогда у нас получится следующее:



Однако результат может быть и не таким, как нам надо. Вот, например, что выйдет, если мы используем формулу $a + b : x + c \times \sin(x)$:



Нужно понимать, что не существует модели без модели. То есть любая модель, которую вы используете, имеет определенную базовую структуру и набор параметров, которые вы можете установить и которые соответствуют вашим данным. И в случае с ChatGPT тоже используется множество параметров (на самом деле их 175 миллиардов).

Примечательно, что лежащая в основе ChatGPT структура (с таким «небольшим» количеством параметров) способна создавать модель, которая вычисляет вероятности следующего слова достаточно хорошо, чтобы у нас получились фрагменты текста размером с настоящее эссе.

МОДЕЛИ ДЛЯ ЧЕЛОВЕЧЕСКИХ ЗАДАЧ

Пример, который мы привели выше, предполагает создание модели числовых данных, которая, по сути, основана на простой физике и старой доброй прикладной математике. Но для ChatGPT мы должны создать модель текста, который будет подобен тому, что написал бы человек. И для этого у нас нет (по крайней мере, пока) ничего похожего на базовую прикладную математику. Итак, что же это за модель?

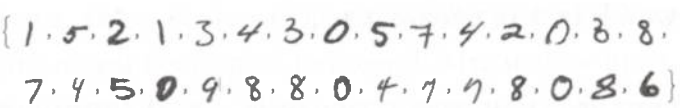
Прежде чем мы поговорим о языке, давайте обсудим другую сугубо человеческую задачу — распознавание образов. И в качестве простого примера предлагаю рассмотреть изображения цифр (это классический пример машинного обучения):



Мы можем взять ряд изображений каждой цифры:



Затем, чтобы выяснить, соответствует ли изображение, полученное нами в качестве входных данных, определенной цифре, мы могли бы сравнить его с имеющимися у нас образцами пиксель за пикселем. Но мы, люди, выполняем эту задачу намного лучше, потому что способны распознать цифры, даже если они написаны от руки и имеют всевозможные модификации и искажения:



Создавая описанную выше модель для наших числовых данных, мы взяли определенное числовое значение x и просто вычислили $a + bx$ для конкретных a и b. В данном случае, если мы определим цвет каждого пикселя как некую переменную x_i, есть ли какая-то функция всех этих переменных, которая — при вычислении — скажет нам, какая цифра изображена в этом случае? Оказывается, такую функцию можно построить. Однако это не так-то просто. И типичный пример может

включать в себя, возможно, полмиллиона математических операций.

Если мы введем весь набор значений пикселей в эту функцию, то на выходе появится число, указывающее, какая цифра изображена на рисунке. Далее мы поговорим о том, как можно сконструировать такую функцию, и о самóм принципе работы нейронных сетей. Но пока давайте рассмотрим функцию как черный ящик, куда мы вводим изображения, скажем, рукописных цифр (в виде массивов значений пикселей), чтобы на выходе получить числа, которым они соответствуют:

```
In[ ]:= NetModel[<...>][{7 0 9 7 8 2 4 1 1 1}]
Out[ ]= {7, 0, 9, 7, 8, 2, 4, 1, 1, 1}
```

Но что на самом деле тут происходит? Допустим, мы постепенно начнем размывать изображение цифры (в нашем примере — цифры 2). Некоторое время наша функция все еще будет распознавать ее. Но вскоре функция начнет выдавать «неправильный» результат:

```
In[ ]:= NetModel[<...>][{2 2 2 2 2 2 2 2 2}]
Out[ ]= {2, 2, 2, 1, 1, 1, 1, 1, 1}
```

Но почему мы рассматриваем этот результат как неправильный? В данном случае мы знаем, что получили

изображения, просто размыв цифру 2. Но если наша цель состоит в том, чтобы создать модель того, как люди распознают изображения, то нужно задать следующий вопрос: что сделал бы человек, не знающий, откуда взялись эти размытые изображения?

Наша модель может считаться хорошей, если результаты, которые выдает наша функция, совпадают с ответами человека. И мы знаем, как создавать функции для распознавания изображений, подобные этой.

Можем ли мы математически доказать, что эти модели работают? В целом — нет. Потому что для этого у нас должно быть математическое описание того, что делаем мы, люди. Возьмите изображение цифры 2 и измените в нем несколько пикселей. Мы можем представить, что с несколькими пикселями не на своих местах мы все равно сумеем считать изображение. Но как долго мы сможем передвигать пиксели и при этом распознавать изображение? Это вопрос зрительного восприятия человека. Ответ, без сомнения, был бы другим для пчел или осьминогов — и совершенно отличался для гипотетических инопланетян.

НЕЙРОННЫЕ СЕТИ

Итак, как же на самом деле работают типичные модели распознавания изображений? В настоящее время самый популярный — и успешный — подход задействует нейронные сети. Изобретенные в 1940-х годах (хотя их форма до сих пор очень близка к современной) нейронные сети можно рассматривать как упрощенное представление того, как, по-видимому, работает наш мозг.

В человеческом мозге находится около 100 миллиардов нейронов (нервных клеток), каждый из которых способен генерировать электрический импульс, возможно, до тысячи раз в секунду. Нейроны соединены в сложную сеть, причем каждый имеет древовидные отростки, позволяющие ему передавать электрические сигналы тысячам других нейронов. Электронный импульс зависит от того, какие импульсы данный нейрон получил от других, при этом разные связи имеют разный вес.

Когда мы видим некое изображение, происходит следующее: фотоны света, отразившиеся от изображения, попадают на особые клетки — фоторецепторы — в задней части глаза, после чего эти клетки передают импульс в виде электрического сигнала нервным клеткам. Эти нервные клетки соединены с другими нейронами, так что в итоге сигналы проходят через целый их ряд. Именно во время этого процесса распознается изображение, и в завершение формируется мысль о том, что мы видим цифру 2 (и, возможно, в конце мы произносим слово «два» вслух).

«Черный ящик» из предыдущего раздела является «математизированной» версией такой нейронной сети. Оказывается, она имеет одиннадцать слоев (четыре из которых считаются основными):

NetChain[

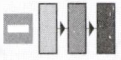
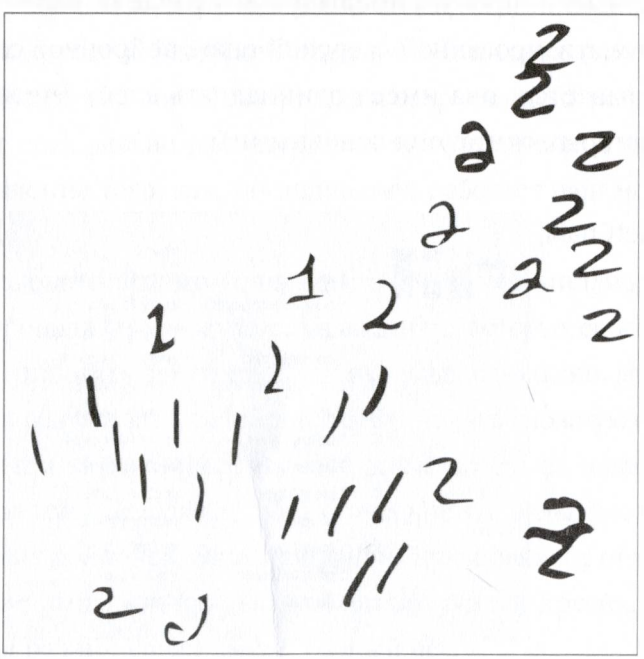


		image
	Input	array (1×28×28)
1	ConvolutionLayer	array (20×24×24)
2	Ramp	array (20×24×24)
3	PoolingLayer	array (20×12×12)
4	ConvolutionLayer	array (50×8×8)
5	Ramp	array (50×8×8)
6	PoolingLayer	array (50×4×4)
7	FlattenLayer	vector (800)
8	LinearLayer	vector (500)
9	Ramp	vector (500)
10	LinearLayer	vector (10)
11	SoftmaxLayer	vector (10)
	Output	class

]

В этой нейронной сети нет ничего теоретически выведенного; все в 1998 году было сконструировано как часть инженерной разработки и оказалось работоспособным. (Это не сильно отличается от ситуации, когда мы описываем мозг как нечто созданное в ходе биологической эволюции.)

Хорошо, но как подобная нейронная сеть распознаёт изображения? Ключевым моментом является понятие аттрактора. Представьте, что у нас есть изображения рукописных цифр 1 и 2:



Мы как бы хотим, чтобы все единицы «притянулись» в одно место, а все двойки — в другое. Иными словами, если изображение больше похоже на единицу, чем на двойку, мы хотим, чтобы оно оказалось единицей, и наоборот. Эти «зоны притяжения» и называются аттракторами (от англ. attract — привлекать, притягивать).

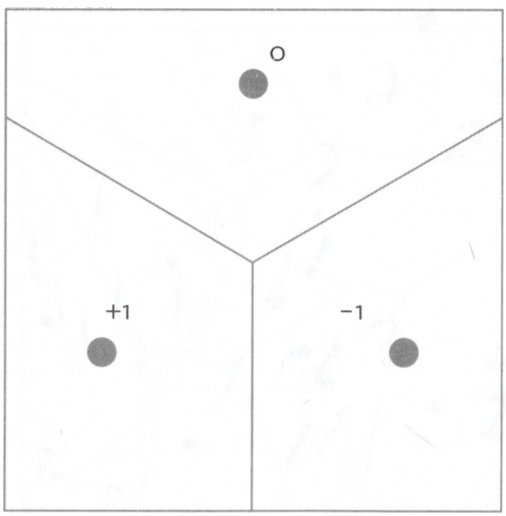
В качестве простой аналогии предположим, что есть определенные позиции на плоскости, отмеченные точками (в реальной жизни это могут быть кафе, обозначенные на карте). Предположим, что, двигаясь из любого места, мы хотим оказаться в ближайшей к нам точке, то есть в ближайшем кафе. Мы можем представить эту плоскость разделенной на области — зоны аттракторов, разграниченные чем-то вроде водоразделов:



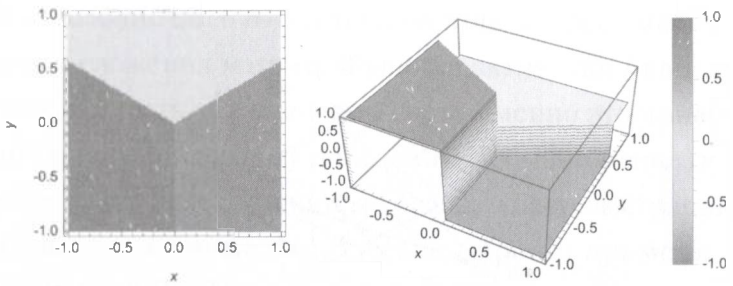
Это можно рассматривать как своего рода задачу на распознавание, где мы не пытаемся понять, на какую цифру похоже изображение, а стараемся увидеть, к какой точке определенная точка находится ближе всего.

(Диаграмма Вороного, которую мы здесь построили, разделяет точки в двумерном евклидовом пространстве; задачу распознавания цифр можно рассматривать как нечто очень похожее, но в 784-мерном пространстве, сформированном из уровней серого всех пикселей на каждом изображении.)

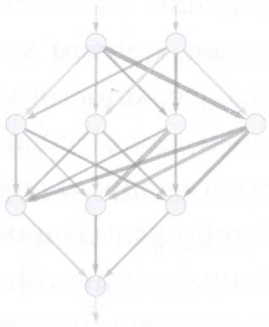
Итак, как мы можем заставить нейронную сеть «распознать» что-то? Давайте рассмотрим такой случай:



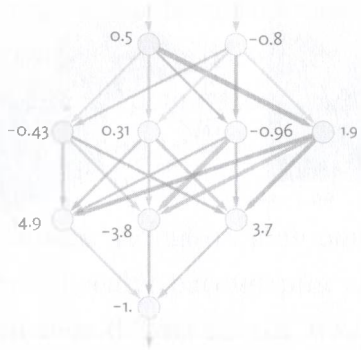
Наша цель состоит в том, чтобы получить входные данные, соответствующие позиции $\{x, y\}$, а затем «распознать» их, то есть отнести к той из трех точек, к которой они находятся ближе всего. Или, другими словами, мы хотим, чтобы нейронная сеть вычисляла функцию от $\{x, y\}$ следующим образом:



Как же мы можем это сделать? В конечном счете нейронная сеть представляет собой связанную совокупность нейронов, обычно расположенных слоями. Простой пример выглядит так:



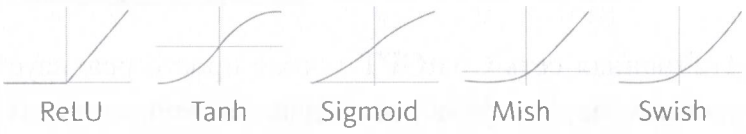
Каждый «нейрон» производит вычисление простой числовой функции. Чтобы задействовать всю сеть, мы просто вводим числа (например, наши координаты x и y) на самом верхнем слое, затем заставляем нейроны каждого следующего слоя вычислять простую числовую функцию и передавать результаты вычисления дальше по сети и в итоге в самом низу получаем:



В традиционной (вдохновленной биологией) установке каждый нейрон фактически имеет определенный набор входящих соединений от нейронов предыдущего уровня, причем каждому соединению присваивается вес, который может быть выражен положительным или отрицательным числом. Значение конкретного нейрона определяется путем умножения значений предыдущих нейронов на их веса, затем сложения результатов, далее добавления константы и, наконец, применения функции

порогового значения (или активации). В математических терминах: если у нейрона есть входные данные $x = \{x_1, x_2 \ldots\}$, то мы вычисляем $f[wx + b]$, где обычно вес w и константа b выбираются поразному для каждого нейрона в сети, а функция f не меняется.

Вычисление $wx + b$ — это всего лишь вопрос умножения и сложения матриц. Функция активации f вводит нелинейность, и в конечном счете именно это приводит к нетривиальному поведению. Обычно используются различные функции активации (на иллюстрации ниже — их примеры). Здесь мы просто применим Ramp (или ReLU):



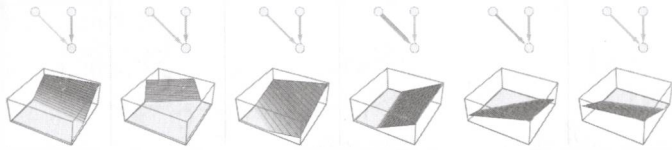
Для каждой задачи, которую нейронная сеть должна выполнить (или, что эквивалентно, для каждой общей функции, которую мы хотим вычислить), у нас будут разные варианты весов. И, как мы обсудим позже, эти веса обычно определяются путем «обучения» нейронной сети на примерах нужных нам выходных данных.

В конечном счете каждая нейронная сеть просто соответствует некоей общей математической функции (хотя ее может быть сложно записать). Для приведенного выше примера это была бы следующая запись:

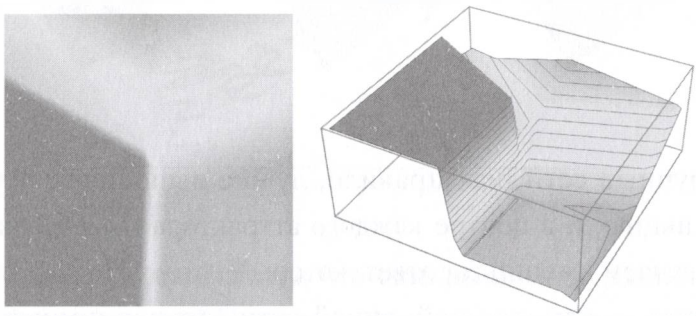
```
w511 f (w311 f (b11 + x w111 + y w112) + w312 f (b12 + x w121 +
    + y w122) + w313 f (b13 + x w131 + y w132) + w314 f (b14 +
    + x w141 + y w142) + b31) +
w512 f (w321 f (b11 + x w111 + y w112) + w322 f (b12 + x w121 +
    + y w122) + w323 f (b13 + x w131 + y w132) + w324 f (b14 +
    + x w141 + y w142) + b32) +
w513 f (w331 f (b11 + x w111 + y w112) + w332 f (b12 + x w121 +
    + y w122) + w333 f (b13 + x w131 + y w132) + w334 f (b14 +
    +  x w141 + y w142) + b33) + b51
```

Нейронная сеть ChatGPT так же просто реализует математическую функцию, подобную этой, но с миллиардами данных.

Однако давайте вернемся к отдельным нейронам. Вот несколько примеров функций, которые нейрон с двумя входными данными (представляющими координаты x и y) может вычислять при различных вариантах весов и констант (и Ramp в качестве функции активации):

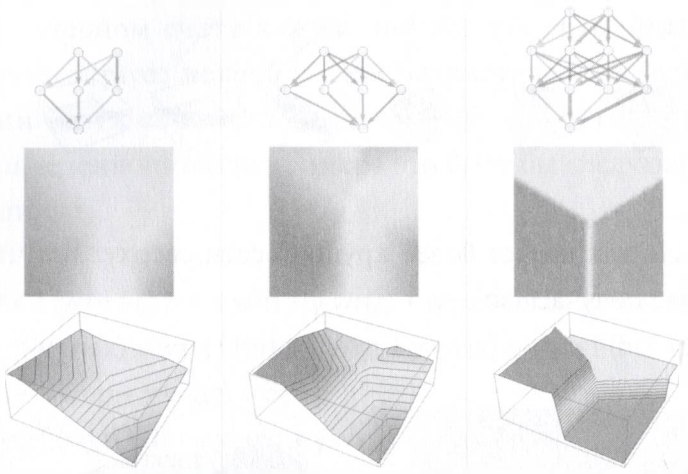


Но как насчет более крупной сети сверху? Вот что у нас получается:



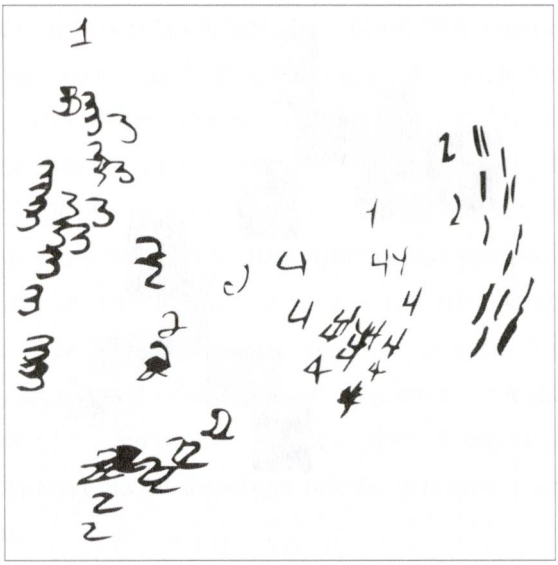
Это не совсем правильно, но близко к функции ближайшей точки, которую мы описали выше.

Давайте посмотрим, что происходит с некоторыми другими нейронными сетями. Как вы узнаете позднее, в каждом случае мы используем машинное обучение, чтобы найти наилучший выбор весов. Здесь мы можем увидеть, что вычисляет нейронная сеть с этими заданными весами:



Крупные сети, как правило, лучше выполняют эту функцию, и в центре каждого аттрактора мы обычно получаем именно тот ответ, который хотим. Но на границах — там, где нейронной сети «трудно принять решение», — ситуация может быть еще более запутанной.

При помощи простой с математической точки зрения задачи на распознавание мы получаем «правильный» ответ. Но в задаче на распознавание рукописных цифр все не так однозначно. Что, если кто-то написал цифру 2 так плохо, что она выглядит как 7? Тем не менее мы можем задаться вопросом, как нейронная сеть различает цифры, и вот как это может выглядеть:



Можем ли мы сказать, что сеть вычисляет различия математически? Не совсем. Математические вычисления — это то, что лежит в основе работы нейронной сети. Но оказывается, что способ того, как нейронная сеть находит различия, очень схож с тем, как это делаем мы, люди.

Давайте возьмем более сложный пример. Допустим, у нас есть изображения кошек и собак и нейронная сеть, которая была обучена различать их. Вот что сеть могла бы сделать с этими изображениями:

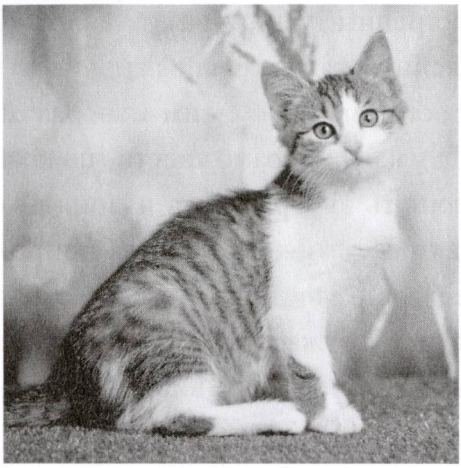


Теперь стало еще непонятнее, какой ответ правильный. А что насчет собаки, одетой в кошачий костюм? Какие бы входные данные ни вводились, нейронная сеть генерирует ответ. И, как выясняется, она делает это таким образом, что ее ответ очень похож на тот, который мог бы дать человек. Как уже упоминалось выше, мы не можем логически вывести этот факт из теоретических положений. Это просто то, что эмпирически было признано верным, по крайней мере в определенных областях. Но это ключевая причина, по которой нейронные сети полезны: они каким-то образом воспроизводят человеческие действия.

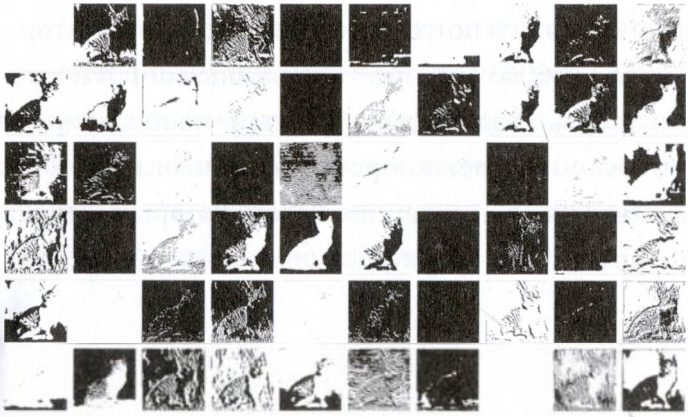
Возьмите фотографию кошки и спросите самого себя: «Почему это кошка?» Возможно, вы начнете так: «Ну, я вижу заостренные уши,» — и так далее. Не очень-то легко объяснить, как вы распознали на изображении кошку. Просто ваш мозг каким-то образом догадался об этом. Но у нас нет возможности (по крайней мере, пока) заглянуть внутрь мозга и понять, как он это сделал. Как насчет искусственной нейронной сети? Что ж, несложно увидеть, что делает каждый нейрон, когда вы загружаете в сеть фотографию кошки. Но получить даже базовую визуализацию уже очень сложно.

В сети выше, которую мы использовали для нахождения ближайшей точки, насчитывается 17 нейронов. В сети для распознавания рукописных цифр 2190 нейронов. А в сети, которую мы используем для распознавания кошек и собак, их 60 650. В обычной жизни было бы довольно сложно визуализировать то, что составляет 60650-мерное пространство. Но поскольку эта сеть создана для работы с изображениями, то многие из слоев нейронов организованы в массивы, подобные массивам пикселей, которые сеть визуализирует.

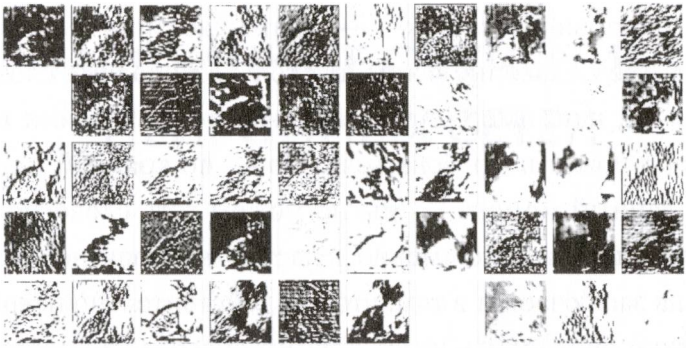
Возьмем типичное изображение кошки:



Мы можем представить состояние нейронов на первом слое с помощью набора производных изображений, многие из которых легко интерпретировать как «кошка без фона» или «контур кошки»:



На десятом слое уже довольно тяжело интерпретировать картинку:



Но в целом мы могли бы сказать, что нейронная сеть «выделяет некоторые черты» (возможно, среди них есть и заостренные уши) и использует их для определения того, что находится на изображении. Однако можно спросить, те же ли это черты, которые выделяем мы, люди? По большей части нет.

Использует ли наш мозг аналогичные функции? Мы этого не знаем. Но примечательно, что первые несколько слоев нейронной сети, подобной той, которую мы показываем здесь, похоже, выделяют аспекты изображений (например, края объектов), которые кажутся похожими на те, которые, как известно, наш мозг выделяет первыми при обработке визуальной информации.

Но давайте предположим, что нам нужно создать «теорию распознавания кошек» нейронными сетями. Мы можем сказать: «Смотрите, целая отдельная сеть только распознает кошек», — и это дает нам некоторое представление о том, насколько проблема сложная (и, например, о том, сколько нейронов или их слоев может потребоваться). Но — по крайней мере на данный момент — мы не можем дать словесного описания того, что именно делает сеть. Вероятно, причина заключается в том, что это действительно невозможно определить и нет другого способа узнать, что делает сеть, кроме как просто проследить за каждым этапом ее работы. Или, может быть, дело в том, что мы еще не разобрались, как работают нейросети, и не определили те законы, которые ими управляют.

Мы столкнемся с такими же проблемами, когда будем говорить о создании языка с помощью ChatGPT. И снова неясно, существуют ли способы обобщить то, что делает сеть. Но богатство и детализация языка (и наш опыт работы с ним) могут позволить нам продвинуться дальше, чем у нас это получилось с изображениями.

МАШИННОЕ ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ

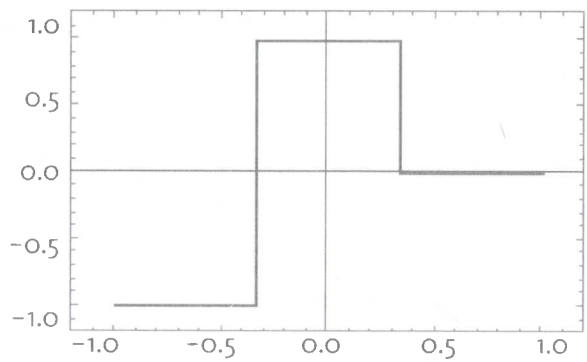
Ранее мы говорили о нейронных сетях, которые «уже знают», как выполнять определенные задачи. Но что делает нейронные сети настолько полезными, так это способность не только выполнять всевозможные задачи, но и постепенно обучаться на различных примерах (в этом нейросети очень похожи на нас).

Когда мы создаем нейронную сеть, способную отличать кошек от собак, нам не нужно писать программу, которая, скажем, обнаруживает усы. Вместо этого мы просто показываем множество примеров того, что такое кошка и что такое собака, а затем заставляем сеть «учиться» на этих примерах.

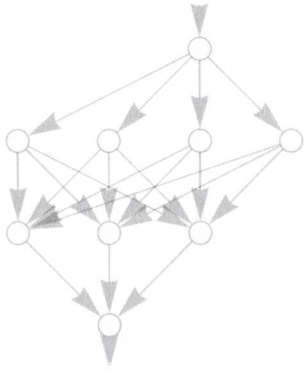
Обученная сеть «обобщает» приметы кошек и собак, исходя из конкретных примеров, которые мы ей даем. Как мы видели выше, дело не просто в том, что сеть распознаёт конкретный пиксельный рисунок изображения кошки. Скорее, нейронная сеть как-то умудряется различать изображения на основе того, что мы, люди, считаем общими кошачьими приметами.

Итак, как же на самом деле происходит обучение нейронной сети? По сути, то, что мы пытаемся сделать, — это найти веса, которые позволят нейронной сети успешно воспроизводить примеры, которые мы ей дали. И затем нейронная сеть интерполирует (обобщает) эти примеры «разумным» способом.

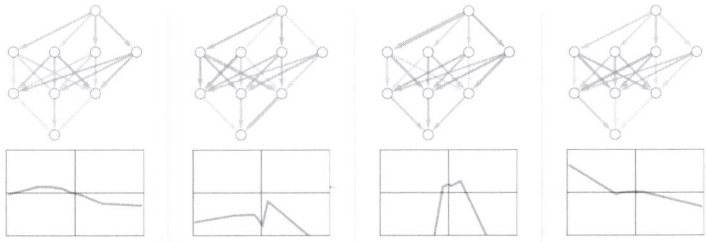
Давайте рассмотрим простой пример. Наша задача — заставить нейронную сеть изучить эту функцию:



Для этой задачи нам понадобится сеть, которая имеет только один вход и один выход, как показано здесь:



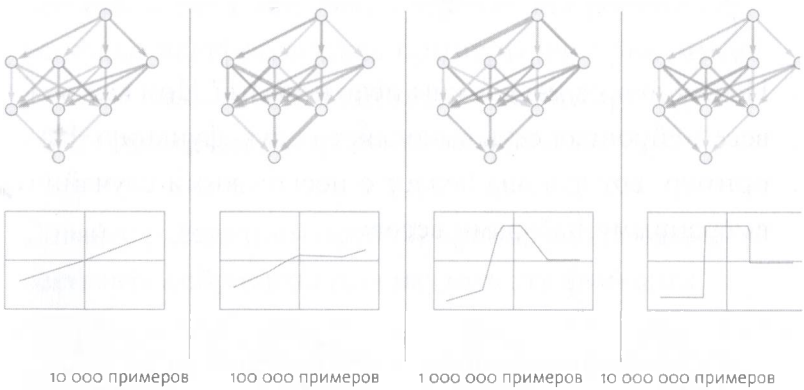
Но какие веса мы должны использовать? При каждом весе нейронная сеть вычисляет некую функцию. Например, вот что она делает с несколькими случайно выбранными наборами весов:



Как мы видим, ни в одном из этих случаев сеть даже близко не воспроизводит нужную нам функцию. Итак,

как нам найти веса, которые будут воспроизводить заданную функцию?

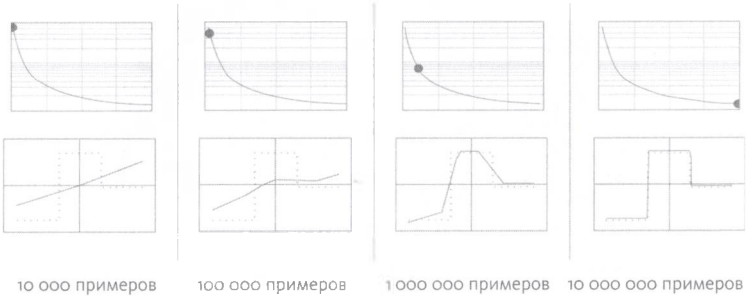
Основная идея состоит в том, чтобы предоставить множество примеров ввода/вывода, на которых можно «учиться», а затем попытаться найти веса, которые будут их воспроизводить. Вот результат, который мы получим, если будем постепенно увеличивать количество примеров для обучения сети:



На каждом этапе этого обучения веса в сети постепенно корректируются, и мы видим, что в итоге получается сеть, которая успешно воспроизводит нужную функцию. Итак, как нам отрегулировать вес? Основная идея заключается в том, чтобы на каждом этапе проверять, насколько мы далеки от получения желаемой

функции, а затем обновлять веса таким образом, чтобы приблизиться к ней.

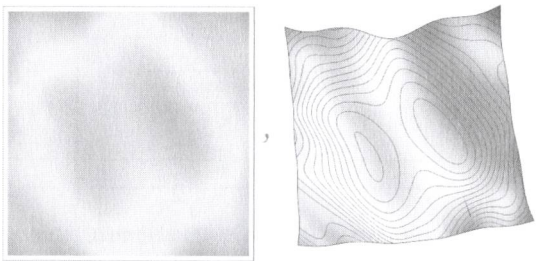
Для того чтобы выяснить, насколько далеко находимся от желаемой функции, мы вычисляем то, что обычно называется функцией потерь (или функцией стоимости). Здесь мы используем простую функцию потерь (L2), которая представляет собой сумму квадратов разностей между полученными и истинными значениями. Мы видим, что по мере продвижения процесса обучения функция потерь постепенно уменьшается (следуя кривой обучения, которая различается для разных задач), пока мы не достигнем точки, где сеть (по крайней мере, в хорошем приближении) успешно воспроизводит нужную нам функцию:



Последняя важная часть, которую необходимо объяснить, — это корректировка весов для уменьшения

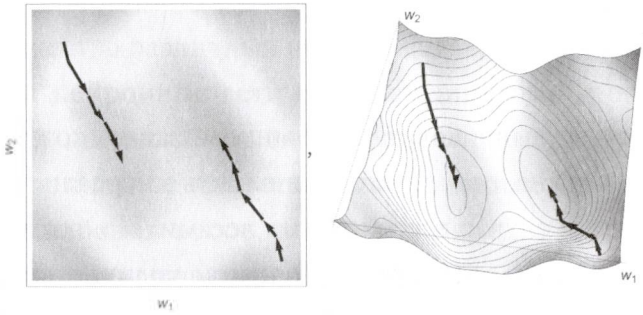
функции потерь. Как мы уже говорили, функция потерь дает нам «расстояние» между полученными и истинными значениями. Но полученные значения на каждом этапе определяются текущей версией нейронной сети и весами в ней. А теперь представьте, что веса являются переменными — скажем, w_i. Мы хотим выяснить, как скорректировать значения этих переменных, чтобы минимизировать потери, которые от них зависят.

Например, предположим, что у нас есть только два веса: w_1 и w_2 (это очень упрощенная нейронная сеть по сравнению с теми, которые используются на практике). Тогда у нас могла бы быть потеря, которая в зависимости от w_1 и w_2 выглядит следующим образом:



Численный анализ предоставляет множество методов для нахождения минимума в подобных случаях. Но наиболее распространенный подход заключается в том,

чтобы просто постепенно следовать по пути самого крутого спуска от любых предыдущих w_1 и w_2, которые у нас были:



Подобно воде, стекающей с горы, эта процедура закончится на некотором локальном минимуме (так сказать, в горном озере), и вполне возможно, что мы так и не достигнем глобального минимума.

Не всегда очевидно, где находится самый крутой спуск на подобном весовом ландшафте. Но здесь нам на помощь приходит математический анализ. Как уже упоминалось выше, мы можем представить нейронную сеть как вычисление математической функции, которая зависит от ее входных данных и их весов. Но теперь давайте рассмотрим дифференциацию в зависимости от этих весов. Оказывается, цепное правило математического анализа позволяет нам распутывать

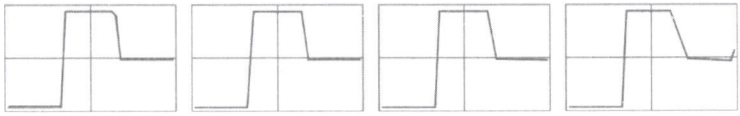
операции, выполняемые последовательными слоями нейронной сети. И в результате мы можем — по крайней мере в некотором локальном приближении — инвертировать работу нейронной сети и постепенно находить веса, которые минимизируют потери выходных данных.

На рисунке выше показан вид минимизации, который нам, возможно, потребуется выполнить в нереалистично простом случае всего с двумя весами. Но оказывается, что даже при гораздо большем количестве весов (ChatGPT использует 175 миллиардов) все еще возможно выполнить минимизацию, по крайней мере до некоторого уровня приближения. И на самом деле большой прорыв в «глубоком обучении», произошедший примерно в 2012 году, был связан с открытием того, что в каком-то смысле легче выполнить минимизацию (по крайней мере, приблизительно), когда задействовано много весов, чем когда их мало.

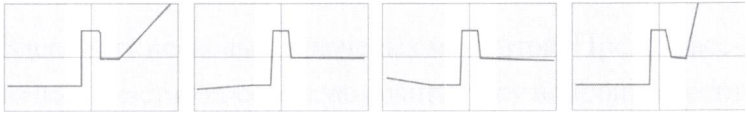
Другими словами — несколько нелогично, — с помощью нейронных сетей легче решать более сложные задачи, чем простые. И причина, по-видимому, заключается в том, что, когда мы имеем много весовых переменных, у нас есть многомерное пространство со множеством направлений, которые могут привести нас к минимуму, тогда как при меньшем

количестве переменных легче застрять в локальном минимуме («горном озере») без возможности из него выбраться.

Стоит отметить, что в типичных случаях существует множество различных наборов весов, которые составляют нейронные сети почти с одинаковой производительностью. И обычно при практическом обучении нейронной сети составляется множество случайных наборов весов, которые приводят к разным, но эквивалентным решениям, подобным представленным ниже:



Однако каждое «другое решение» будет иметь немного иное поведение. И если мы попросим нейросеть, скажем, экстраполировать его за пределы области, в границах которой приводили обучающие примеры, то можем получить совершенно разные результаты:



Но какой из них «правильный»? На самом деле сказать сложно. Все эти результаты «согласуются с наблюдаемыми данными», но все соответствуют разным «врожденным» способам нейронной сети «думать о том», что значит выполнять задачу «нестандартно». И некоторые из них могут казаться нам, людям, «более разумными», чем другие.

ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ: ПРАКТИКА И ЗНАНИЯ

В течение последнего десятилетия мы наблюдали множество достижений в искусстве обучения нейронных сетей. Да, по сути, это самое настоящее искусство. Иногда — особенно в ретроспективе — можно увидеть проблеск «научного объяснения» того, что происходило с ИИ. Однако почти все результаты были достигнуты благодаря методу проб и ошибок: ученые дорабатывали первоначальные идеи и ухищрялись как только могли, что позволило им постепенно накопить знания о том, как работать с нейронными сетями.

Здесь есть несколько ключевых моментов. Прежде всего возникает вопрос, какую архитектуру нейронной сети следует использовать для конкретной задачи. Следом

встает вопрос, как получить данные, на основе которых можно обучить нейронную сеть. Сегодня мы всё чаще имеем дело с обучением сети не с нуля: новая сеть может либо напрямую включать в себя другую, уже обученную сеть, либо использовать ее для создания большого количества примеров для собственного обучения.

Можно предположить, что для каждого вида задач требуется разная архитектура нейронной сети. Но ученые обнаружили, что одна и та же архитектура, по-видимому, подходит даже для совершенно разных, казалось бы, задач. На каком-то уровне это напоминает идею универсальных вычислений (и мой принцип вычислительной эквивалентности), но, как расскажу позже, я думаю, это скорее отражение того факта, что задачи, которые мы обычно даем нейронным сетям, являются «человекоподобными», и нейронные сети могут улавливать довольно общие «человекоподобные процессы».

Когда нейронные сети только появились, существовала точка зрения, что нужно заставлять их делать как можно меньше. Например, при преобразовании речи в текст считалось, что сначала следует проанализировать аудиозапись речи, разбить ее на фонемы и так далее. Но было обнаружено, что — по крайней мере для «человекоподобных» задач — обычно лучше просто попытаться обучить нейронную сеть «сквозной»

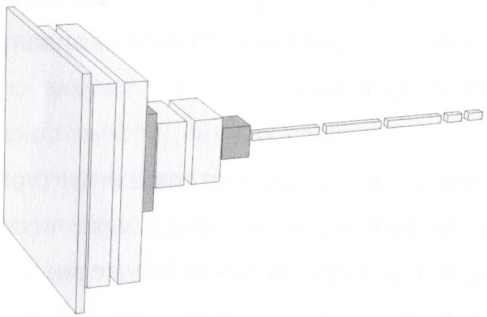
задаче, позволяя сети обнаружить необходимые промежуточные характеристики, кодировки и прочее самостоятельно.

Была также идея, что в нейронную сеть следует вводить отдельные сложные компоненты, чтобы она могла фактически реализовывать определенные алгоритмические идеи. Но опять же, по большей части это оказалось бесполезным. Вместо этого лучше просто взять очень простые компоненты и позволить им «самоорганизоваться» (способами, которых мы не можем понять), чтобы достичь (предположительно) эквивалента этих алгоритмических идей.

Это не значит, что не существует идей структурирования, которые могли бы помочь при использовании нейронных сетей. Так, например, наличие двумерных массивов нейронов с локальными связями кажется очень полезным на ранних стадиях обработки изображений. И структуры связи, которые концентрируются на последовательном обращении к прошлым данным, как мы увидим позже, тоже выглядят полезными при работе с чем-то вроде языка, как в ChatGPT.

Но важной особенностью нейронных сетей является то, что они, как и компьютеры в целом, в конечном счете просто работают с данными. А современные нейронные

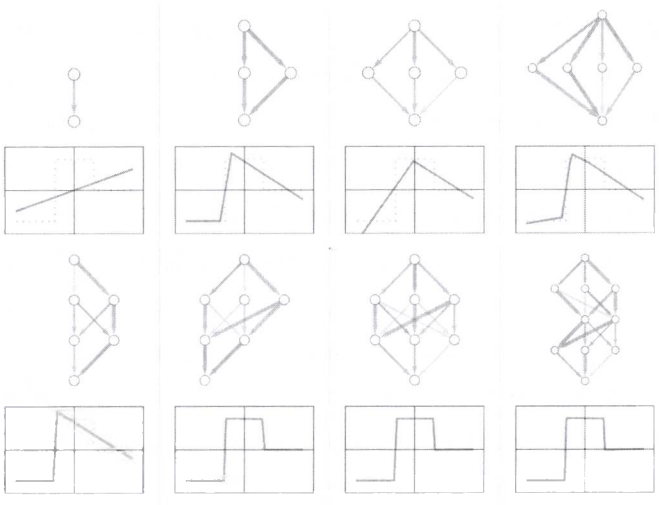
сети — с современными подходами к обучению нейросетей — работают с массивами чисел. В процессе обработки эти массивы могут быть полностью изменены или реорганизованы. Например, сеть, которую выше мы использовали для идентификации цифр, начинается с двумерного массива, представляющего изображение и быстро сгущающегося до множества каналов, но затем концентрирующегося в одномерный массив, который в итоге и будет содержать элементы, представляющие возможные цифры:



Хорошо, но как можно определить, насколько большая нейронная сеть понадобится для выполнения конкретной задачи? Это целое искусство. В каком-то смысле главное здесь — самим понимать, насколько сложна поставленная задача. Но задачи, которые обычно решает человек, очень трудно оценить. Да, может существовать систематический способ выполнения их механически с помощью компьютера. Но трудно

сказать, есть ли кратчайший путь, который позволит выполнять задачу, по крайней мере на человеческом уровне, значительно легче. Возможно, потребуется пройти много уровней, чтобы механически сыграть в определенную игру. Но, возможно, существует гораздо более простой (эвристический) способ добиться «игры на человеческом уровне».

Когда мы имеем дело с крошечными нейронными сетями и простыми задачами, иногда можно отчетливо увидеть, что добраться из одной точки в другую просто невозможно. Например, вот как может выглядеть выполнение задачи из предыдущего раздела с помощью нескольких небольших нейронных сетей:



Мы видим, что если сеть слишком мала, то она просто не сможет воспроизвести нужную нам функцию. Но если сеть больше определенного размера, то проблем быть не должно — по крайней мере если тренировать ее довольно долго и на достаточном количестве примеров. Кстати говоря, эти картинки иллюстрируют часть наших знаний о нейронных сетях: нередко можно обойтись сетью меньшего размера, если в середине есть определенное «сжатие», которое заставляет информацию проходить через меньшее промежуточное число нейронов. (Также стоит упомянуть, что сети без промежуточного уровня — так называемые перцептронные — могут изучать только линейные функции. Но как только появляется хотя бы один промежуточный уровень, всегда можно произвольно выстроить любую функцию, по крайней мере если у сети достаточно нейронов. Однако для того, чтобы сделать их реально обучаемыми, обычно требуется своего рода регуляризация или нормализация.)

Итак, давайте предположим, что вы остановились на определенной архитектуре нейронной сети. Теперь возникает проблема получения данных для обучения сети. И многие практические задачи, связанные с нейронными сетями и машинным обучением в целом, сосредоточены на получении или подготовке необходимых обучающих данных. Во многих случаях

требуется получить примеры входных данных и ожидаемых результатов (так называемое контролируемое обучение). Например, изображения можно переименовывать в зависимости от того, что на них, или снабжать каким-то атрибутом. Возможно, кому-то придется этим заняться, для чего понадобится много усилий. Однако нередко можно брать что-то готовое, применяя это как своего рода прокси-элемент. Например, можно использовать Alt-теги, которыми уже были помечены изображения в интернете. Также можно задействовать субтитры, которые были созданы для видео. Для обучения языковому переводу применимы уже переведенные на разные языки версии веб-страниц или других документов.

Сколько данных нужно дать нейронной сети, чтобы обучить ее выполнению конкретной задачи? Это трудно определить. Конечно, количество данных можно значительно снизить, если использовать так называемое трансферное обучение — перенос данных, которые уже были изучены в другой сети. Но, как правило, нейронным сетям нужно много примеров, чтобы обучение проходило успешно. И для некоторых задач также важно, чтобы примеры были повторяющимися. Действительно, это стандартная стратегия — просто снова и снова показывать нейронной сети все имеющиеся примеры. В каждой из этих эпох обучения нейронная

сеть будет находиться в разном состоянии и каким-то образом напоминать самой себе о конкретном примере, чтобы запомнить его. (Возможно, это аналогично тому, как повторение материала помогает человеку запомнить информацию.)

Но нередко просто повторения одного и того же примера недостаточно: нейронной сети необходимо показать вариации. И, как уже было выяснено, для того, чтобы быть полезными, эти вариации не обязательно должны быть сложными. Простое незначительное изменение изображений может создать эффект новизны. Или, например, если заканчиваются реальные видео, скажем, по обучению вождению беспилотных автомобилей, можно просто взять данные из симуляций в модельной среде, подобной видеоиграм, без деталей, свойственных реальным видеозаписям.

А что же ChatGPT? У него есть приятная особенность, которая заключается в способности выполнять обучение без присмотра, а это значительно облегчает поиск примеров. Напомним, основная задача ChatGPT состоит в том, чтобы выяснить, как продолжить фрагмент текста, который ему был предоставлен. Итак, чтобы получить обучающие примеры, все, что нужно сделать, — это взять фрагмент текста и замаскировать его конец, а затем использовать этот фрагмент

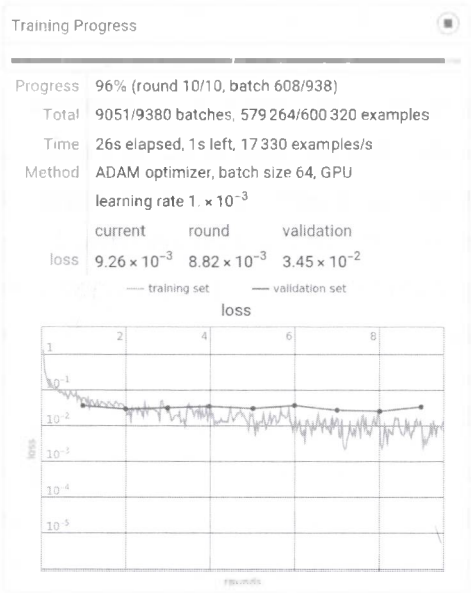
в качестве входных данных для обучения; при этом выходными данными будет полный фрагмент текста. Позже мы обсудим это подробнее, но главное, что нам нужно знать, — это то, что, скажем, в отличие от распознавания изображений здесь нам не нужны явные теги. ChatGPT может учиться на любых примерах текста, которые ему даны.

А что насчет самого процесса обучения в нейронной сети? В конце концов, все дело сводится к определению того, какие веса наилучшим образом будут соответствовать приведенным примерам. Существуют всевозможные подробно описанные варианты выбора и настройки гиперпараметров (веса допустимо рассматривать как параметры), которые можно использовать. Имеются различные варианты функции потерь (сумма квадратов, сумма абсолютных значений и так далее). Есть различные способы минимизации потерь (как далеко можно перемещаться с каждым шагом и так далее). И тогда возникают вопросы вроде того, сколько примеров нужно показать, чтобы получить каждую последующую оценку потерь, которые человек пытается минимизировать. И кстати, можно применить машинное обучение (как мы делаем, например, в Wolfram Language*)

* Wolfram — язык программирования, разработанный компанией Wolfram Research для системы Mathematica. *Прим. пер.*

для автоматизации машинного обучения — и для автоматической установки таких вещей, как гиперпараметры.

В конце концов, весь процесс обучения можно охарактеризовать следующим образом, наблюдая, как постепенно уменьшаются потери (как на этом мониторе прогресса Wolfram Language для небольшого тренинга):



Мы можем увидеть, что потери на некоторое время уменьшаются, но в итоге выравниваются до постоянного значения. Если это значение небольшое,

то обучение можно считать успешным. В противном случае это, вероятно, признак того, что следует попробовать изменить архитектуру сети.

Можно ли с уверенностью предсказать, сколько времени должно пройти, чтобы кривая обучения выровнялась? Здесь, как и везде, существуют приблизительные степенные соотношения масштабирования, которые зависят от размера нейронной сети и объема используемых данных. Однако общий вывод таков: обучение нейронной сети — дело сложное и требует больших вычислительных усилий. И с практической точки зрения подавляющее большинство этих усилий тратится на выполнение операций с массивами чисел, с чем хорошо справляются графические процессоры, — вот почему обучение нейронных сетей обычно зависит от доступности графических процессоров.

Появятся ли в будущем принципиально иные и более эффективные способы обучать нейронные сети или делать то, что они сейчас делают? Я думаю, что, скорее всего, да. Фундаментальная идея нейронных сетей заключается в создании гибкой вычислительной структуры, состоящей из большого числа простых (по сути, идентичных) компонентов, которую можно постепенно модифицировать, чтобы она училась на примерах.

В современных нейронных сетях для такой постепенной модификации используются идеи математического анализа, применяемые к действительным числам. Но становится все более очевидным, что числа с высокой точностью не столь важны: восемь или даже меньше бит достаточно даже при использовании современных методов.

Когда речь идет о вычислительных системах вроде клеточных автоматов, которые параллельно работают со множеством отдельных битов, не совсем ясно, как выполнить подобную постепенную модификацию, однако нет причин думать, что это невозможно. И вполне вероятно, что, как и в случае с прорывом в области глубокого обучения в 2012 году, такие постепенные модификации будут проще в более сложных случаях.

Немного похожие на человеческий мозг, нейросети имеют фиксированную сеть нейронов, в которой изменяется только сила взаимодействия между ними — их вес. Возможно, в мозге молодого организма и может возникнуть значительное количество совершенно новых связей. Однако даже если с точки зрения биологии это имеет смысл, то не совсем ясно, каков лучший способ достижения необходимой нам функциональности. И то, что включает в себя эквивалент прогрессивного переписывания сети (возможно,

напоминающее наш проект по физике), вполне может в конечном счете оказаться лучше.

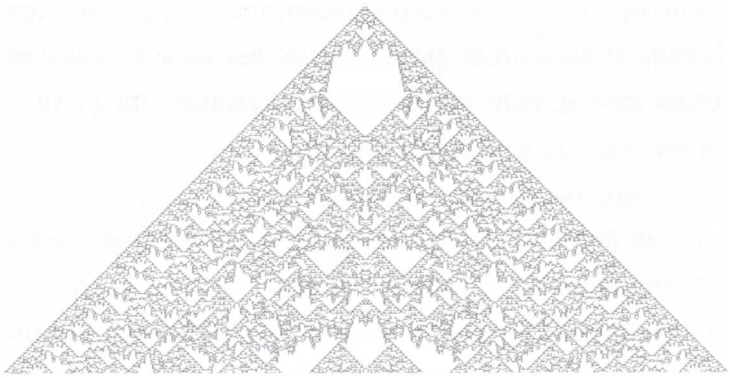
Но даже в отношении существующих нейронных сетей в настоящее время есть важное ограничение: в том виде, как оно проводится сейчас, обучение нейросети принципиально является последовательным, при этом после каждой эпохи обучения примеры возвращаются обратно, чтобы можно было обновить вес нейронов. И действительно, при возможностях современного компьютерного оборудования, даже с учетом графических процессоров, значительное количество нейронной сети бóльшую часть времени обучения простаивает, при этом обновляется только одна ее часть за раз. И в некотором смысле это происходит потому, что в наших современных компьютерах память отделена от процессора (и от графического в том числе). Но в мозге, по-видимому, все по-другому: каждый элемент памяти (например, нейрон) также является потенциально активным вычислительным элементом. И если бы мы могли настроить наше будущее компьютерное оборудование схожим образом, обучение нейросетей могло бы быть гораздо эффективнее.

ДОСТАТОЧНО БОЛЬШАЯ СЕТЬ МОЖЕТ СДЕЛАТЬ ВСЕ ЧТО УГОДНО — ТАК ВЕДЬ?

Возможности программ наподобие ChatGPT выглядят впечатляюще. Можно даже подумать, что если продолжать тренировать нейронные сети, то в конце концов они научатся делать все. И когда речь идет о вещах, которые легко доступны человеческому мышлению, то вполне возможно, что все обстоит именно так. Но урок, который можно вынести из последних нескольких сотен лет развития науки, заключается в том, что есть вещи, которые могут быть поняты с помощью формальных процессов, но недоступны человеческому мышлению.

Одним из ярких примеров является нетривиальная математика. Но самое важное — это вычисления.

И в конечном счете проблема заключается в феномене вычислительной несводимости. Есть вычисления, для выполнения которых, как можно было бы подумать, потребуется много шагов, но которые на самом деле могут быть сведены к чему-то совершенно простому. Однако открытие вычислительной несводимости подразумевает, что это не всегда работает. Так, существуют процессы (вероятно, подобные показанному ниже), в которых для определения того, что происходит, требуется отслеживать каждый шаг вычислений:



Знания о том, как работает наш мозг, предположительно позволяют нам избежать вычислительной несводимости. Чтобы вычислять в уме, требуются особые усилия, и на практике невозможно продумать этапы работы любой нетривиальной программы.

Конечно, для этого у нас есть компьютеры. А с помощью компьютеров мы можем легко выполнять длительные, не поддающиеся вычислению задачи. И ключевым моментом является то, что здесь, как правило, нет кратчайшего пути.

Да, мы можем запомнить множество конкретных примеров из какой-то вычислительной системы. И возможно, даже можем увидеть некоторые (вычислительно сводимые) закономерности, которые позволят нам сделать небольшое обобщение. Но дело в том, что вычислительная несводимость означает невозможность гарантировать, что непредвиденное не произойдет. И, только выполнив вычисления, вы можете сказать, что на самом деле происходит в каждом конкретном случае.

В конце концов, существует фундаментальное противоречие между обучаемостью и вычислительной несводимостью. Обучение, по сути, предполагает сжатие данных за счет использования закономерностей. Но вычислительная несводимость подразумевает, что в конечном счете есть предел количеству существующих закономерностей.

С практической точки зрения можно представить превращение небольших вычислительных устройств — таких,

как клеточные автоматы или машины Тьюринга, — в обучаемые системы вроде нейронных сетей. И действительно, такие устройства могут служить хорошими инструментами для нейронной сети: например, Wolfram|Alpha может быть хорошим инструментом для ChatGPT. Но вычислительная несводимость подразумевает, что нельзя «проникнуть» внутрь этих устройств и заставить их обучаться.

Другими словами, существует окончательный компромисс между возможностями и обучаемостью: чем сильнее вы хотите, чтобы система по-настоящему использовала свои вычислительные возможности, тем больше она будет демонстрировать вычислительную несводимость и тем меньше поддаваться обучению. А чем легче что-то поддается фундаментальному обучению, тем меньше оно способно выполнять сложные вычисления.

(Для ChatGPT в том виде, в каком он существует сейчас, ситуация еще более экстремальная, потому что нейронная сеть, используемая для генерации каждого токена вывода, является чистой нейронной сетью с прямой связью — feed-forward, то есть сетью без циклов, — и, следовательно, не имеет возможности выполнять какие-либо вычисления с нетривиальным «потоком управления».)

Конечно, можно спросить, действительно ли важно уметь выполнять несводимые вычисления. На протяжении большей части человеческой истории это не было так уж важно. Но современный технологичный мир построен на инженерии, которая использует математические вычисления, а также — все чаще — более общие вычисления. Если мы посмотрим на мир природы, то увидим, что он полон несводимых вычислений, и мы постепенно начинаем понимать, как их имитировать и использовать в технологических целях.

Нейронная сеть, безусловно, способна замечать те виды закономерностей в мире природы, которые и мы можем легко заметить с помощью «невооруженного человеческого мышления». Но если мы хотим разобраться в вещах, которые относятся к математике или вычислительной науке, то нейронная сеть не сможет этого сделать, если только не использует в качестве инструмента «обычную» вычислительную систему.

Во всем этом есть что-то непонятное. Раньше существовало множество задач, включая написание текстов, которые считались фундаментально слишком сложными для компьютеров. И сегодня, когда мы видим, что технологии вроде ChatGPT справляются с этими задачами, мы склонны думать, что все дело в том, что компьютеры стали значительно мощнее — в частности

тех, которые были способны делать что-то вроде вычисления поведения вычислительных систем, таких как клеточные автоматы.

Но это не совсем правильный вывод. Вычислительно несводимые процессы по-прежнему являются таковыми и по-прежнему принципиально сложны для компьютеров — даже если компьютеры могут легко вычислить отдельные этапы. Вывод, который мы должны сделать, заключается в том, что задачи (например, написание текстов), которые мы могли выполнять, но считали, что компьютеру они не под силу, на самом деле с точки зрения вычислений проще, чем мы предполагали.

То есть причина, по которой нейронная сеть может хорошо писать тексты, заключается в том, что написание текстов оказывается более простой с вычислительной точки зрения задачей, чем мы думали. И в некотором смысле это приближает нас к формулированию теории о том, как нам, людям, удается создавать тексты и в целом работать с языком.

Если бы у нас была достаточно большая нейронная сеть, она могла бы делать все, что под силу простым людям. Но она бы все равно не смогла охватить все то, на что способен мир природы в целом — или что могут

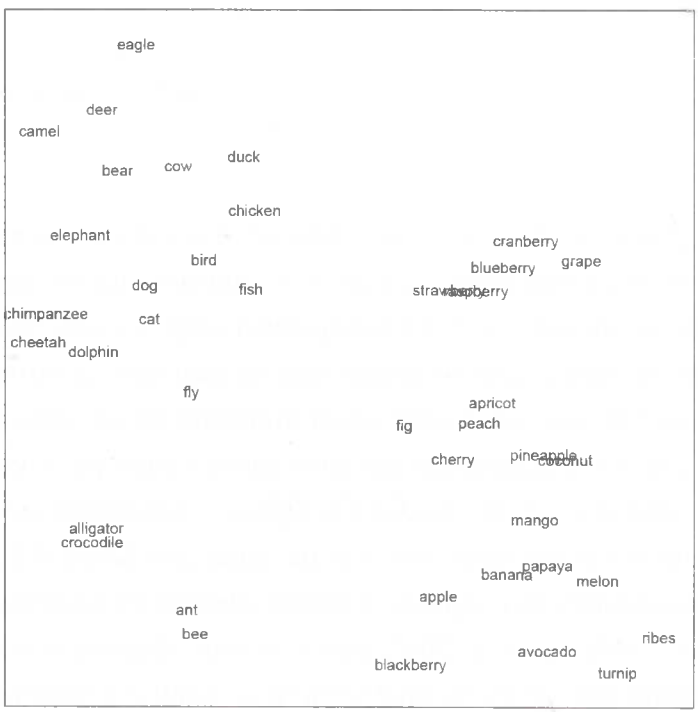
делать инструменты, которые мы создали, опираясь на законы природы. Именно применение этих инструментов — как практических, так и концептуальных — позволило нам за последние столетия выйти за границы того, что доступно чистой человеческой мысли без посторонней помощи, и использовать для человеческих целей больше того, что существует в физической и вычислительной вселенной.

КОНЦЕПЦИЯ ЭМБЕДДИНГОВ

Нейронные сети — по крайней мере в том виде, в каком они существуют сейчас, — основаны на числах. Следовательно, если мы собираемся использовать сети для работы с чем-то вроде текста, нам понадобится представить текст в виде чисел. Конечно, можно начать с простого присвоения каждому слову в словаре номера (как это делает ChatGPT). Однако существует концепция (и она является центральной для ChatGPT), которая выходит за данные рамки. Это так называемая идея эмбеддингов. Эмбеддинг можно объяснить как способ представить сущность чего-либо с помощью массива чисел, где подобные объекты представлены близлежащими числами.

Так, например, мы можем объяснить эмбеддинг как попытку разместить слова в своего рода пространстве значений, где близкие по значению слова находятся

рядом. Реальные эмбеддинги, которые используются, скажем, в ChatGPT, состоят из больших списков чисел. Но если мы спроецируем это в двумерную модель, то вот как это будет выглядеть:



То, что мы здесь видим, вполне соответствует нашей картине мира. Но как создать подобный эмбеддинг? Грубо говоря, идея состоит в том, чтобы просмотреть большие объемы текста (в данном случае

пять миллиардов слов из интернета), а затем выяснить, насколько похожи контексты разных слов. Так, например, «аллигатор» и «крокодил» будут встречаться в похожих предложениях, и это означает, что они окажутся рядом друг с другом в эмбеддингах. Напротив, «репа» и «орел», как правило, не встречаются в одном предложении, поэтому в эмбеддинге они будут расположены далеко друг от друга.

Но как реализовать нечто подобное с помощью нейронных сетей? Давайте начнем с того, что поговорим об эмбеддингах не для слов, а для изображений. Мы хотим найти способ охарактеризовать изображения с помощью списков чисел таким образом, чтобы изображениям, которые мы считаем похожими, присваивались похожие списки чисел.

Как можно определить, какие изображения похожи друг на друга? Если они состоят, скажем, из рукописных цифр, мы могли бы считать два изображения похожими, если на них изображена одна и та же цифра. Ранее мы говорили о нейросети, которая обучена распознавать цифры, написанные от руки. Она может быть настроена таким образом, чтобы в качестве конечного результата распределять изображения в десять различных групп — по одной для каждой цифры.

Но что, если мы «перехватим» те процессы, которые происходят внутри нейронной сети, до того, как сеть примет окончательное решение, что это, скажем, цифра 4? Мы могли бы ожидать, что внутри нейронной сети есть числа, которые характеризуют изображения как «в основном похожие на 4, но немного похожие на 2» или что-то в этом роде. И идея состоит в том, чтобы подобрать такие числа для использования в качестве элементов при эмбеддинге.

Итак, концепция будет заключаться в следующем: вместо того чтобы пытаться понять, какое изображение похоже на другое, мы рассматриваем четко поставленную задачу (в данном случае распознавание цифр), для которой можем получить четкие обучающие данные, и при выполнении этой задачи нейронная сеть должна принять неявное решение о схожести изображений. Таким образом, мы уже не говорим о близости изображений, вместо этого мы обсуждаем конкретный вопрос о том, какая цифра соответствует изображению, а затем оставляем нейронной сети определить, что подразумевается под близостью изображений.

Как же будет работать сеть при распознавании цифр? Мы можем представить, что сеть состоит из 11 последовательных слоев, которые можно символически

изобразить следующим образом (функции активации показаны в виде отдельных слоев):



Вначале мы загружаем в первый слой наши изображения, представленные двумерными массивами значений пикселей. В конце — из последнего слоя — мы получаем массив данных из десяти значений, которые сообщают нам, насколько сеть уверена в том, что изображение соответствует каждой из цифр от 0 до 9.

Когда мы вводим изображение *4*, значения нейронов в последнем слое будут следующие.

$\{1{,}42071 \times 10^{-22}, 7{,}69857 \times 10^{-14}, 1{,}9653 \times 10^{-16}, 5{,}55229 \times 10^{-21}, 1.,$
$8{,}33841 \times 10^{-14}, 6{,}89742 \times 10^{-17}, 6{,}52282 \times 10^{-19}, 6{,}51465 \times 10^{-12},$
$1{,}97509 \times 10^{-14}\}$

Другими словами, нейронная сеть к этому моменту однозначно «уверена» в том, что это изображение цифры 4, и, чтобы действительно получить результат «4», нам просто нужно выбрать позицию нейрона с наибольшим значением.

Но что, если мы сделаем еще один шаг? Самая последняя операция в сети — это так называемый softmax, который пытается форсировать определенность. До того как мы применили эту операцию, значения нейронов были следующими:

{−26,134, −6,02347, −11,994, −22,4684, 24,1717, −5,94363, −13,0411, −17,7021, −1,58528, −7,38389}

Нейрон, представляющий цифру 4, по-прежнему имеет наибольшее числовое значение. Но информация также есть и в других нейронах. И мы можем ожидать, что этот список чисел может быть использован для характеристики сути изображения и таким образом дать нам что-то, что мы можем использовать уже в качестве эмбеддинга. Например, приведенные ниже четверки мало отличаются друг от друга, однако все они сильно отличаются от восьмерок:



Здесь мы, по сути, десятью числами характеризуем наши изображения. Но нередко лучше использовать больше. Например, в нашей сети распознавания цифр мы можем получить массив из 500 чисел, перейдя

на предыдущий слой. И это, вероятно, разумный размер массива для эмбеддинга изображений.

Если мы хотим создать визуализацию пространства изображений для рукописных цифр, то должны уменьшить размерность и спроецировать полученный 500-мерный вектор в трехмерное пространство:



Выше мы говорили о создании характеристики (и, следовательно, эмбеддинга) изображений, основанной на выявлении сходства изображения путем определения их соответствия (согласно нашему обучающему набору) одной и той же рукописной цифре. Мы можем сделать то же самое для гораздо более широкого спектра изображений, если у нас есть обучающий набор, который определяет, скажем, к какому из 5000 распространенных типов объектов (кошка, собака, стул и так далее) относится каждое изображение. Таким образом мы

способны создать эмбеддинг изображения, который привязан к определенной категории, но затем «обобщается» в соответствии с поведением нейронной сети. Дело в том, что такое поведение согласуется с тем, как мы, люди, воспринимаем и интерпретируем изображения. В конечном счете этот эмбеддинг покажется нам правильным и полезным при выполнении «человеческих» задач.

Итак, как по такому же принципу найти эмбеддинг слов? Главное — постановка задачи, в соответствии с которой мы сможет обучить сеть. Стандартной задачей в данном случае будет «подсказывание слов». Представьте, что у нас есть словосочетание «the ___ cat [*англ.* кошка]». Если опираться на большой объем текста (скажем, в текстовом содержимом интернета), то какие слова с высокой вероятностью могли бы заполнить пробел? Или, например, какая вероятность может быть у сопутствующих слов в словосочетании «___ black [*англ.* черная] ___»?

Как описать эту задачу нейронной сети? Мы должны сформулировать ее в цифрах. Один из способов — присвоить уникальный номер каждому из примерно 50 тысяч наиболее распространенных слов в английском языке. Так, например, артикль the может получить номер 914, а слово cat (с пробелом перед ним) — 3542. (Это фактические числа, используемые GPT-2.) Итак, для

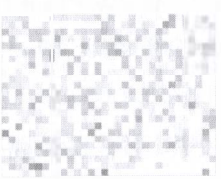
задачи «the ___ cat» наши входные данные будут {914, 3542}. Каким окажется результат? Ну, это должен быть список из 50 тысяч или около того чисел, которые дают вероятности для каждого из возможных «заполняющих» слов. Опять-таки, чтобы найти эмбеддинг, мы хотим «перехватить» состояние нейронной сети непосредственно перед тем, как она «примет решение», а затем получить список чисел, которые там встречаются и которые мы можем рассматривать как характеризующие каждое слово.

Как же выглядят эти характеристики? За последние десять лет была разработана последовательность различных систем (word2vec, GloVe, BERT, GPT…), каждая из которых основана на уникальном подходе к нейронным сетям. Но в конечном счете все они берут слова и характеризуют их списками из сотен или тысяч цифр.

В необработанном виде эти векторы эмбеддинга довольно неинформативны. Например, вот что выдает GPT-2 в качестве необработанных векторов эмбеддинга для трех конкретных слов:



кошка



собака



стул

Если мы измерим расстояние между этими векторами, то сможем определить близость этих слов к другим словам. Позже мы более подробно обсудим когнитивные значения подобных эмбеддингов. Но на данный момент для нас самое главное заключается в том, что мы можем превращать слова в наборы чисел, которые нейронная сеть способна понимать.

На самом деле мы можем характеризовать наборами чисел не только слова, но и последовательности слов или даже целые блоки текста. Именно так и работает ChatGPT. Он берет готовый текст и генерирует вектор эмбеддинга. Затем высчитывает вероятности последующих слов. Далее генерирует список чисел, которые, по сути, дают вероятности для каждого из 50 тысяч или около того возможных слов.

Строго говоря, ChatGPT работает не со словами, а с токенами — удобными лингвистическими единицами, которые представляют собой слова или фрагменты слов вроде pre, ing или ised*. Использование токенов облегчает обработку редких и сложных слов, а также слов из других языков, а иногда позволяет — хорошо это или плохо? — изобретать новые слова.

* В английском языке эти части слов образуют разные формы глаголов. *Прим. пер.*

ВНУТРИ CHATGPT

Теперь мы наконец можем перейти к тому, как именно работает ChatGPT, что происходит у него внутри. ChatGPT представляет собой гигантскую нейронную сеть. В настоящее время это версия так называемой сети GPT-3, в которой насчитывается 175 миллиардов весов. Эта нейронная сеть во многих отношениях похожа на другие, которые мы обсуждали ранее. Отличием является то, что она специально настроена для работы с языком. Самой примечательной ее особенностью является часть архитектуры, которая называется трансформером.

Как мы уже обсуждали выше, в первых нейронных сетях каждый нейрон на каждом слое связан с каждым нейроном предыдущего слоя. Но подобная связь (предположительно) является излишней, если вы работаете с данными, которые имеют определенную, известную

структуру. Так, например, на ранних стадиях работы с изображениями обычно используются так называемые сверточные нейронные сети (convnets), в которых нейроны расположены на сетке, аналогичной пикселям на изображении, и связаны только с близлежащими к ним нейронами.

Идея трансформера состоит в том, чтобы сделать нечто похожее и для последовательностей токенов, из которых состоит фрагмент текста. Но вместо того, чтобы определять фиксированную область в последовательности, где могут быть связи, трансформеры вводят понятие «внимание», то есть некоторым частям текста уделяется больше внимания, чем другим. Возможно, однажды можно будет просто запустить универсальную нейронную сеть и выполнить всю настройку с помощью обучения. Но по крайней мере на данный момент кажется, что на практике очень важно модулировать вещи — так, как это делают трансформеры и, вероятно, человеческий мозг.

Итак, что же на самом деле делает ChatGPT (а скорее сеть GPT-3, которая лежит в его основе)? Напомним, что его главная задача заключается в том, чтобы продолжить текст «разумным» способом, основываясь на тех «знаниях», которые были получены во время обучения (заключавшегося в просмотре миллиардов

страниц текста из интернета и прочих источников). То есть в любой момент времени у ChatGPT есть определенный объем текста, и главная задача состоит в том, чтобы «придумать» следующий подходящий вариант токена.

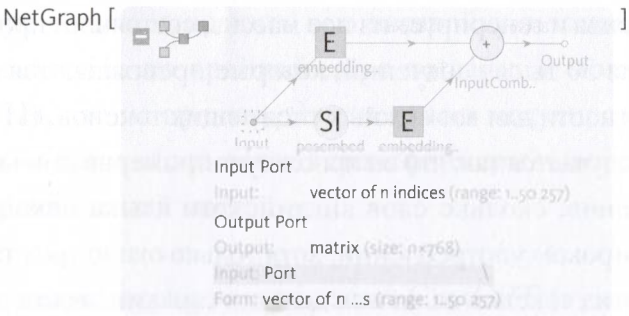
ChatGPT работает в три этапа. Сначала берет последовательность токенов, соответствующую тексту на данный момент, и находит эмбеддинг (то есть массив чисел), который представляет эти токены. Затем продолжает работу над этим эмбеддингом при помощи стандартного подхода нейронной сети, когда значения «просачиваются» через последовательные слои сети для создания нового эмбеддинга (то есть нового массива чисел). После ChatGPT берет последнюю часть этого массива и генерирует из нее массив, состоящий примерно из 50 тысяч значений, которые превращаются в вероятности для возможных следующих токенов. (И иногда случается так, что он использует примерно столько же токенов, сколько слов английского языка находится в широком употреблении, хотя только около трех тысяч из этих токенов являются целыми словами, а остальные представляют собой фрагменты слов.)

Важно отметить, что каждая часть этого конвейера реализована нейронной сетью, веса которой определяются путем сквозного обучения сети. Другими

словами, в действительности ничто, кроме общей архитектуры, не является «явно спроектированным»: все просто «изучается» на основе входных данных.

Архитектура нейронной сети содержит множество деталей, отражающих всевозможный опыт работы нейронных сетей и знания о них. И хотя это выходит за рамки повествования, я думаю, полезно поговорить о некоторых из этих деталей, чтобы получить представление о том, что входит в нейросети, подобные ChatGPT.

Сначала идет модуль эмбеддинга. Это его схематическое представление на языке Wolfram Language для GPT-2:

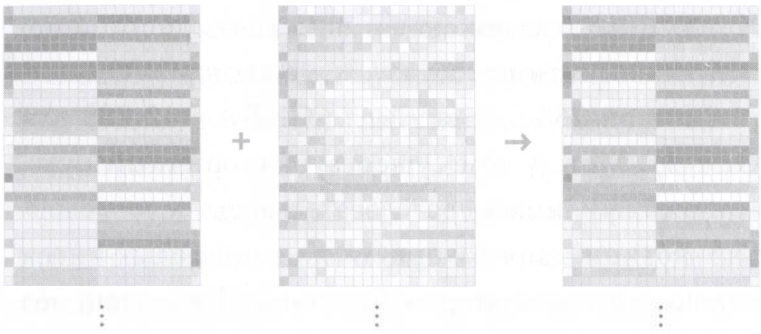


В качестве входных данных мы имеем вектор из n токенов (представленных, как и в предыдущем разделе, целыми числами от 1 примерно до 50 000).

Каждый из этих токенов с помощью однослойной нейронной сети преобразуется в вектор эмбеддинга (длиной 768 значений для GPT-2 и 12 288 для GPT-3). Между тем существует вторичный путь, который принимает последовательность целочисленных позиций для токенов и из этих целых чисел создает другой вектор эмбеддинга. И наконец, оба вектора эмбеддинга — из значения и из позиции токена — складываются вместе, чтобы получилась окончательная последовательность векторов эмбеддинга из этого модуля.

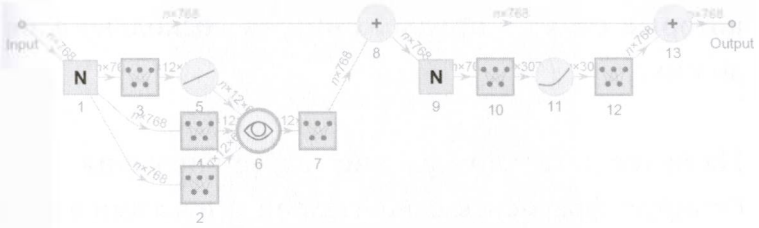
Для чего нам нужно складывать векторы эмбеддинга значения и эмбеддинга положения токена? Здесь нет никакой научно обоснованной причины. Путем проб и ошибок мы пришли к тому, что срабатывает именно такой метод. И это то, что нужно знать о нейронных сетях: до тех пор, пока настройки будут «примерно правильными», достаточно просто провести полноценное обучение, без необходимости понимания на глубоком уровне, как работает нейронная сеть и как она сама себя настраивает.

Вот результат работы модуля эмбеддинга для строки «hello hello hello hello hello hello hello hello hello hello bye bye bye bye bye bye bye bye bye bye»:



Выше показаны элементы вектора эмбеддинга для каждого токена, слева направо изображена последовательность эмбеддингов слова «hello», а затем последовательность эмбеддингов слова «bye». Второй массив представляет собой позиционный эмбеддинг, в котором структура выглядит очень рандомно, и это отличный пример «случайно выученного» материала (здесь в GPT-2).

Итак, после модуля эмбеддинга трансформер выстраивает последовательность так называемых блоков внимания (12 для GPT-2, 96 для GPT-3). Все это довольно сложно и напоминает типичные труднодоступные для понимания инженерные или, если уж на то пошло, биологические системы. Вот, например, схематическое изображение одного блока внимания (для GPT-2):

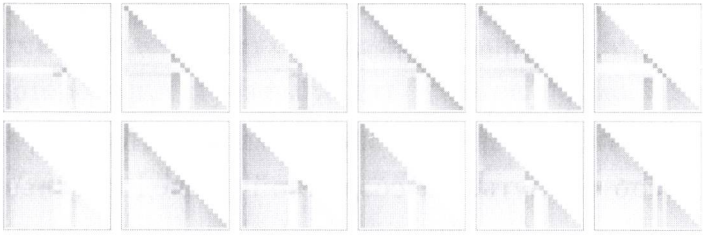


Внутри каждого такого блока внимания есть набор так называемых голов внимания (*англ.* attention heads) — 12 штук для GPT-2 и 96 для GPT-3, — каждая из которых работает независимо друг от друга с разными фрагментами значений в векторе эмбеддинга. (При этом мы не знаем, почему разделение вектора эмбеддинга было хорошей идеей и что означают разные его части, — это просто работает.)

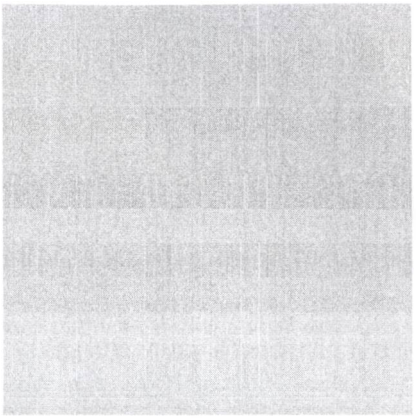
Итак, что же такое головы внимания? По сути, это способ вернуться к созданной последовательности токенов (то есть к тексту) и упаковать готовый текст в ту форму, которая будет полезна для поиска следующего токена. В первом разделе мы говорили об использовании вероятностей для выбора пар слов на основе слов, которые им предшествовали. Механизм внимания в трансформере позволяет смотреть глубже, на гораздо более ранние слова, таким образом потенциально улавливая, как, скажем, глаголы могут относиться к существительным,

которые стоят в предложении за несколько слов до них.

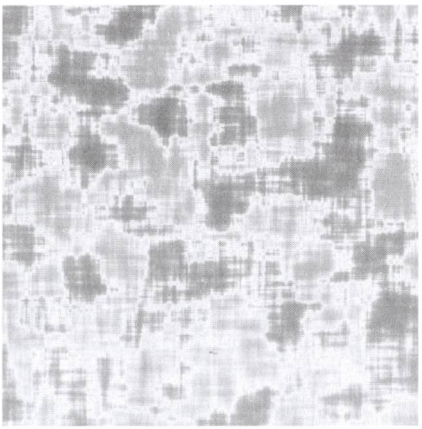
На более детальном уровне голова внимания комбинирует фрагменты с определенными весами в векторах эмбеддинга, связанными с различными токенами. Так, например, 12 голов внимания в первом блоке внимания (в GPT-2) имеют следующие шаблоны весов рекомбинации («оглянись-в-начало-последовательности-токенов») для строки «hello... bye», приведенной выше:



После обработки головой внимания полученный повторно взвешенный вектор эмбеддинга (длиной 768 значений для GPT-2 и 12 288 для GPT-3) передается через стандартный «полностью подключенный» слой нейронной сети. Нам трудно разобраться в том, что делает этот слой. Но вот ниже представлен график матрицы весов размером 768 × 768, которую он использует (здесь он изображен для GPT-2):



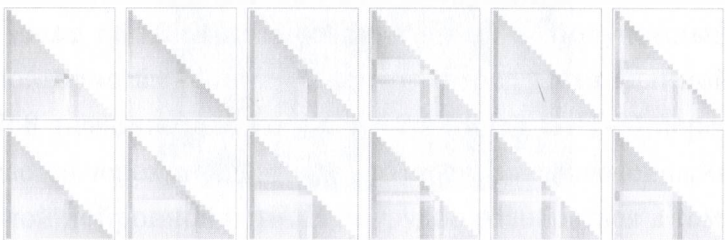
Если принять во внимание скользящие средние* 64 × 64, то начинает вырисовываться немного рандомная структура:



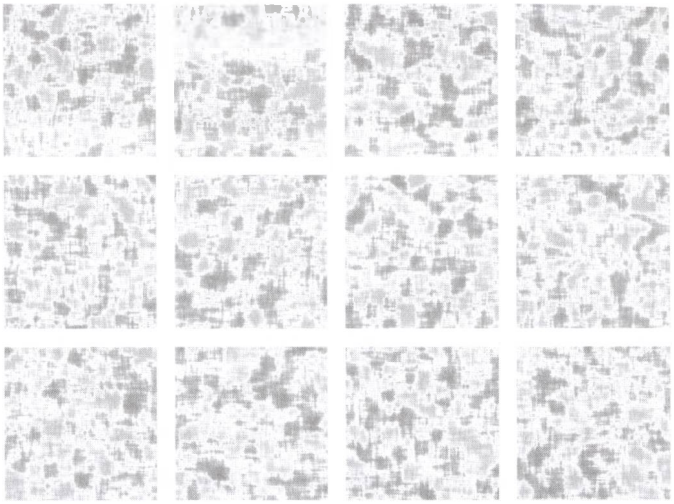
* Скользящая средняя — общее название для семейства функций, значения которых в каждой точке определения равны некоторому среднему значению исходной функции за предыдущий период. *Прим. пер.*

Что определяет эту структуру? Скорее всего, это некое «нейросетевое кодирование» особенностей человеческого языка. Но на данный момент нам пока неизвестно, что это могут быть за особенности. По факту мы как будто вскрыли мозг ChatGPT (или, по крайней мере, GPT-2) и обнаружили, что здесь все очень сложно и ничего толком непонятно, даже если самому ChatGPT удается создавать узнаваемый человеческий язык.

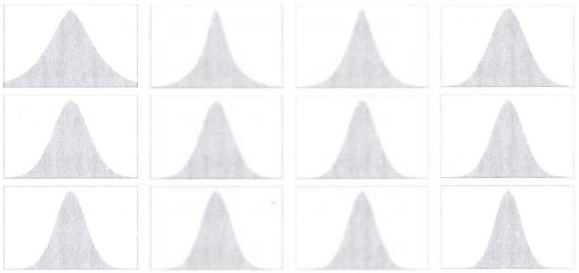
Итак, пройдя через один блок внимания, мы получаем новый вектор эмбеддинга, который затем последовательно проходит через дополнительные блоки (всего 12 блоков для GPT-2, 96 для GPT-3). Каждый блок имеет собственный особый паттерн внимания и полностью подключенных весов. Так выглядит последовательность весов внимания первого блока GPT-2 для строки «hello... bye»:



А вот скользящие средние матрицы для полностью связанных слоев:



Любопытно, что, хотя эти матрицы весов в разных блоках внимания выглядят довольно похоже, распределение размеров весов может несколько отличаться (и не всегда происходит по Гауссу):



Итак, что делает трансформер после прохождения всех этих блоков внимания? По сути, он преобразовывает исходную коллекцию эмбеддингов для последовательности

токенов в конечную коллекцию. Затем ChatGPT выбирает последний эмбеддинг и «расшифровывает» его, чтобы получить список вероятностей следующего токена.

Вот в общих чертах то, что происходит внутри ChatGPT. Все это может показаться сложным (не в последнюю очередь из-за множества несколько произвольных «инженерных решений»), но на самом деле задействованные базовые элементы удивительно просты. Потому что, в конце концов, мы имеем дело лишь с нейронной сетью, состоящей из искусственных нейронов, каждый из которых выполняет простую операцию по получению набора числовых входных данных, а затем комбинирует их с определенными весами.

Исходными входными данными для ChatGPT является массив чисел (векторы эмбеддинга для токенов). И когда ChatGPT создает новый токен, эти числа «пульсируют» по слоям нейронной сети, при этом каждый нейрон отвечает за свою задачу и передает результат нейронам на следующем слое. Здесь нет зацикливания или откатов назад, все просто передается дальше по сети.

Этот метод сильно отличается от работы типичной вычислительной системы вроде машины Тьюринга,

в которой результаты многократно перерабатываются одними и теми же вычислительными элементами. Здесь — по крайней мере при генерации заданного токена вывода — каждый вычислительный элемент (нейрон) задействован только один раз.

Но в некотором смысле даже в ChatGPT существует внешний цикл, который повторно использует вычислительные элементы. Потому что, когда ChatGPT собирается сгенерировать новый токен, он всегда считывает (принимает в качестве входных данных) всю последовательность токенов, которые ему предшествуют, включая те, которые ранее записал сам ChatGPT. Именно поэтому мы можем считать, что ChatGPT действительно — по крайней мере на самом внешнем уровне — включает в себя цикл обратной связи, хотя и такой, в котором каждая итерация явно видна в виде токена, появляющегося в генерируемом им тексте.

Но давайте вернемся к сути: ChatGPT — это нейронная сеть, которая многократно используется для генерации каждого токена. На каком-то уровне это коллекция идентичных искусственных нейронов. Некоторые части сети состоят из всецело связанных между собой слоев нейронов, где каждый нейрон на конкретном слое является контрольной копией, соединенной (благодаря

некоторому весу) с каждым нейроном на предыдущем слое. Но благодаря архитектуре на основе трансформеров у ChatGPT есть более структурированные части, где на разных слоях подключены только некоторые нейроны. (Конечно, мы по-прежнему можем сказать, что все нейроны здесь связаны между собой, просто некоторые из них имеют нулевой вес.)

Кроме того, в ChatGPT есть особые аспекты нейронной сети, из-за которых его нельзя рассматривать как сеть, состоящую из однородных слоев. Например, как мы обсуждали ранее, внутри блока внимания есть места, где создается множество копий входящих данных, каждая из которых затем проходит свой путь обработки, потенциально включающий в себя разное количество слоев, и только позже рекомбинируется. Рассматривать процесс таким образом очень удобно, но важно также помнить о плотном заполнении слоев, где некоторые веса равны нулю.

Если посмотреть на самый длинный путь ChatGPT, то в нем задействовано около 400 (основных) уровней — в некотором смысле не так уж много. Но в ChatGPT существуют миллионы нейронов, что дает в общей сложности 175 миллиардов связей и, следовательно, 175 миллиардов весов. И нужно понимать, что каждый раз, когда ChatGPT генерирует новый токен, он выполняет

вычисление, задействующее каждый из этих весов. Реализация данных вычислений может быть организована по слоям в виде высокопараллельных операций с массивами, которые удобно выполнять на графических процессорах. Но для каждого произведенного токена все равно необходимо осуществить 175 миллиардов вычислений (и в конце еще немного), так что неудивительно, что генерация длинного фрагмента текста с помощью ChatGPT может занять некоторое время.

Но наиболее замечательно то, что все эти операции — какими бы простыми они ни были по отдельности — вместе каким-то образом могут выполнять «человеческую» работу по генерированию текста. Следует еще раз подчеркнуть, что (по крайней мере, насколько нам известно) нет никакого окончательного теоретического обоснования, почему все это работает. Так что, я думаю, мы должны рассматривать это как потенциальное научное открытие: каким-то образом в нейронной сети, подобной ChatGPT, возможно уловить суть того, что человеческому мозгу удается делать при работе с языком.

ОБУЧЕНИЕ CHATGPT

Итак, мы в общих чертах рассказали о том, как работает ChatGPT. Но как он настроен? Как были определены все эти 175 миллиардов весов в его нейронной сети? По сути, они являются результатом очень масштабного обучения, основанного на огромном массиве текстов — в интернете, оцифрованных книг и так далее, — написанных людьми. Как мы уже говорили, все эти обучающие данные не гарантируют, что нейронная сеть сможет успешно создавать текст, подобный написанному человеком. Для того чтобы она все же научилась это делать, по-видимому, необходимы новые инженерные разработки. Но самым большим сюрпризом и открытием ChatGPT стало то, что это вообще возможно и что нейронная сеть со «всего лишь» 175 миллиардами весов может создавать разумную модель текста.

Сегодня в цифровом виде нам доступно огромное количество написанных людьми текстов. В Сети насчитывается по меньшей мере несколько миллиардов страниц, написанных человеком, а общий объем этого супертекста составляет, возможно, триллион слов. Если же включить в него непубличные веб-страницы, то объем будет как минимум в сто раз больше. В данный момент у нас есть доступ к пяти с лишним миллионам оцифрованных книг (из 100 миллионов когда-либо написанных), что составляет еще около 100 миллиардов слов. И это даже не учитывая текстов видеоматериалов. (Для сравнения: мной за всю мою жизнь было опубликовано чуть менее 3 миллионов слов, а за последние тридцать лет я написал около 15 миллионов слов в электронных письмах, в общей сложности напечатал около 50 миллионов слов и всего за последние два-три года я произнес более 10 миллионов слов в прямых трансляциях. Вскоре буду обучать бота на основе наработок.)

Как с помощью всех этих текстов можно обучить нейронную сеть? Базовый процесс очень похож на тот, который мы уже обсуждали выше. Вы даете сети набор примеров, а затем корректируете веса, чтобы свести к минимуму ошибки (потери), которые она может допустить в дальнейшем. Самое главное, что при обратном распространении ошибки вы каждый раз меняете вес (хотя бы немного), поэтому вам приходится иметь

дело с большим количеством весов. (Фактические обратные вычисления, как правило, лишь немного сложнее прямых.)

Современные графические процессоры позволяют параллельно вычислять результаты тысяч примеров. Но когда дело доходит до фактического обновления весов в нейронной сети, это должно выполняться постепенно. (И вероятно, именно здесь человеческий мозг, с его комбинированными вычислительными элементами и памятью, на данный момент имеет по крайней мере архитектурное преимущество.)

Даже в случаях изучения числовых функций, кажущихся простыми, которые мы обсуждали ранее, мы обнаружили, что иногда приходится использовать миллионы примеров для успешного обучения сети, по крайней мере при обучении с нуля. Сколько же тогда примеров понадобится для обучения нейросети человеческому языку? Похоже, какого-либо фундаментального теоретического способа узнать это не существует. Но на практике ChatGPT был успешно обучен на нескольких сотнях миллиардов слов текста.

Некоторые части текста использовались для обучения один раз, другие — несколько. Однако каким-то образом ChatGPT все равно «получил то, что ему было

нужно» из текста, который «увидел». Но, учитывая огромный объем текста для изучения, насколько большой должна быть сеть, чтобы хорошо выучить его? Опять же, у нас пока нет теоретического ответа. Как мы обсудим далее, предположительно существует определенное общее алгоритмическое содержание человеческого языка и того, что люди обычно выражают с его помощью. Следующий вопрос заключается в том, насколько эффективной будет нейронная сеть при реализации модели, основанной на этом алгоритмическом содержании. Но, опять-таки, пока мы этого не знаем, хотя успех ChatGPT говорит о том, что данная модель достаточно эффективна.

И в конце мы можем просто отметить, что ChatGPT работает, используя две-три сотни миллиардов весов, что по количеству сопоставимо с общим количеством слов (или токенов), предоставленных ему в обучающих данных. В некотором смысле удивительно (хотя подобное мы наблюдаем и в других похожих на ChatGPT системах), что размер сети сопоставим с размером обучающих данных. Это, конечно, не значит, что весь этот текст из интернета, книг и так далее хранится внутри ChatGPT. Потому что на самом деле внутри ChatGPT находится набор чисел с точностью чуть менее 10 знаков, которые представляют собой своего рода распределенную кодировку совокупной структуры всего этого текста.

Иными словами, мы могли бы спросить, каково же эффективное информационное содержание человеческого языка. Допустим, у нас есть несколько фрагментов необработанного текста и представление этого текста в нейронной сети ChatGPT. Это представление, скорее всего, далеко от алгоритмически минимального представления (об этом мы поговорим далее), но именно его использует нейронная сеть. В этом представлении происходит незначительное сжатие обучающих данных. В среднем для передачи информационного содержания одного слова из обучающего материала требуется чуть меньше одного веса нейронной сети.

Когда мы запускаем ChatGPT для генерации текста, нам, по сути, приходится использовать каждый вес один раз. Таким образом, если у нас есть n весов, нам нужно выполнить порядка n вычислительных шагов, хотя на практике многие из них обычно выполняются параллельно. Но если нам нужно около n слов обучающих данных для настройки этих весов, то (следуя из того, что было сказано выше) нам потребуется около n^2 вычислительных шагов для обучения сети. Вот почему при современных методах приходится говорить о затратах на обучение стоимостью в миллиарды долларов.

ПОМИМО БАЗОВОГО ОБУЧЕНИЯ

Основная часть усилий при обучении ChatGPT тратится на то, чтобы показать ему большое количество существующего текста из интернета, оцифрованных книг и так далее. Но, оказывается, есть и другая — по-видимому, довольно важная — часть обучения.

Как только сеть получит всю сырую информацию на основе исходного массива текста, который был изучен, она готова начать генерировать собственный текст, одновременно продолжая учиться на подсказках. И хотя сгенерированный текст кажется разумным, у ChatGPT есть тенденция — особенно при генерировании длинных фрагментов текста — сильно отклоняться (то есть нередко текст становится непохож на то, что написал бы человек). Это не то, что можно легко обнаружить при традиционном статистическом анализе текста, однако люди точно это заметят.

Ключевая идея, легшая в основу создания ChatGPT, заключалась в том, чтобы позволить реальным людям активно взаимодействовать с ChatGPT, видеть, что он производит, и, по сути, давать ему обратную связь о том, «как быть хорошим чат-ботом». Но как нейронная сеть может использовать этот фидбэк? Во-первых, можно просто попросить людей оценить результаты работы нейронной сети. Затем строится другая модель нейронной сети, которая пытается предсказать эти оценки, после чего эту модель прогнозирования можно запускать — по сути, как функцию потерь — в исходной сети, фактически позволяя настраивать сеть с помощью обратной связи от человека. На практике результаты оказывают большое влияние на успех системы в производстве чего-то человеческого.

Удивительно, как мало изменений нужно внести в первоначально обученную сеть, чтобы заставить ее двигаться в нужном направлении. Можно было бы подумать, что для дальнейшего обучения придется запускать постоянные алгоритмы обучения, корректировать веса и так далее.

Но все не так. Вместо этого, кажется, достаточно что-то сказать ChatGPT один раз — как часть промпта, который вы ему задаете, — и он сможет успешно использовать подсказку при генерации текста. Повторюсь,

тот факт, что это работает, является ключом к пониманию того, что ChatGPT делает и как это связано со структурой человеческого языка и мышления.

В этом определенно есть что-то человеческое: проведя всю предварительную подготовку ChatGPT, вы можете один раз сказать ему что-то, а он запомнит это на достаточное количество времени, чтобы сгенерировать с помощью полученной информации фрагмент текста. Что же происходит в этом случае? Возможно, все, что вы можете сказать, уже заложено в сеть и вы просто задаете ChatGPT правильное направление. Однако это не кажется правдоподобным. Более вероятно, что все элементы уже существуют, но вся конкретика определяется чем-то вроде траектории между ними, то есть задачей, которую вы ставите перед ChatGPT.

Так же, как и в случае с людьми, если сказать ChatGPT что-то странное и неожиданное, совершенно не вписывающееся в известные ему рамки, он не сможет успешно интегрировать эту информацию. Он сумеет это сделать только в том случае, если интегрирует новые данные поверх уже имеющейся структуры.

Также стоит еще раз отметить, что у нейронной сети существуют так называемые алгоритмические пределы того, что сеть может уловить. Если дать сети

поверхностные правила вроде «это принадлежит этому» и так далее, то нейронная сеть, скорее всего, сможет это воспроизвести: то, что сеть уже почерпнула из нашего языка, дает ей отличный пример для подражания. Но если вы попросите ChatGPT провести настоящие «глубокие» вычисления, которые включают в себя множество шагов, то это просто не сработает. (Помните, что на каждом этапе ChatGPT всегда пересылает данные в свою сеть, никогда не зацикливаясь, за исключением случаев генерации новых токенов.)

Конечно, сеть может узнать ответ для конкретных несводимых вычислений. Но как только появляются вероятности, этот подход не срабатывает. Итак, пришло время протянуть нейронным сетям руку помощи и использовать настоящие вычислительные инструменты. (Для этого однозначно подходят Wolfram|Alpha и Wolfram Language, потому что они созданы для того, чтобы «говорить о вещах в мире», точно так же как и нейронные сети языковой модели.)

ЧТО ПОЗВОЛЯЕТ CHATGPT РАБОТАТЬ?

Человеческий язык и процесс мышления — это своего рода последний уровень сложности. Удивительно, что все это функционирует благодаря мозгу, состоящему всего лишь из 100 миллиардов (или около того) нейронов (и, возможно, из 100 триллионов связей). Можно предположить, что в мозге находятся не только нейроны, а целый мир, который физика еще не познала. Но теперь благодаря ChatGPT мы знаем, что простая искусственная нейронная сеть, имеющая примерно столько же связей, сколько нейронов в мозге человека, способна на удивление хорошо генерировать человеческий язык.

Да, это по-прежнему большая и сложная система: в нейронной сети столько же весов, сколько в мире на сегодняшний день написано слов. Однако нам еще

сложно поверить, что богатство языка и все сложные вещи, о которых мы говорим на этом языке, могут быть заключены в такую ограниченную систему. Частично причиной этого становится повсеместное явление, которое впервые стало очевидным на примере «правила тридцати»: вычислительные процессы могут существенно увеличивать кажущуюся сложность систем, даже если их основные правила просты. Но на самом деле, как мы уже говорили ранее, нейронные сети, используемые в ChatGPT, сконструированы таким образом, чтобы минимизировать этот эффект (и связанную с ним вычислительную несводимость), что делает обучение более доступным.

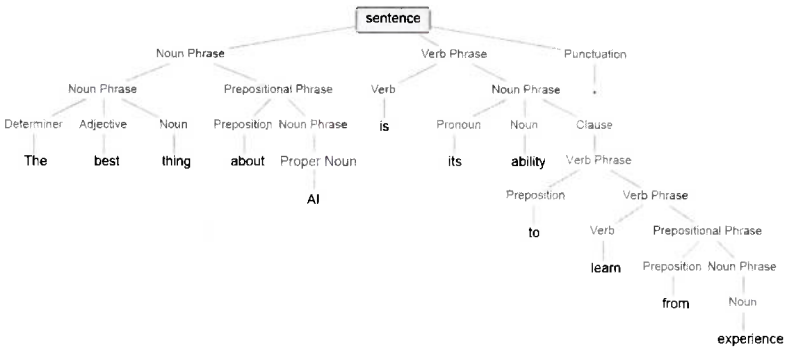
Как же тогда системы вроде ChatGPT добиваются прогресса в языковой области? Я думаю, ответ заключается в том, что на фундаментальном уровне язык проще, чем кажется. А это означает, что ChatGPT — даже с его предельно простой структурой нейронной сети — успешно способен уловить суть и логику человеческого языка. Более того, в процессе обучения ChatGPT каким-то образом «неявно обнаружил» те закономерности в языке (и мышлении), которые делают это возможным.

Я думаю, что успех ChatGPT позволяет ожидать открытия новых масштабных законов языка — и, по сути,

законов мышления. В ChatGPT, который построен как обычная нейронная сеть, эти законы непременно работают, хотя мы их и не знаем. Но если бы мы могли каким-то образом выявить данные законы, то смогли бы сделать то, что ChatGPT уже делает, гораздо более простыми, эффективными и прозрачными способами.

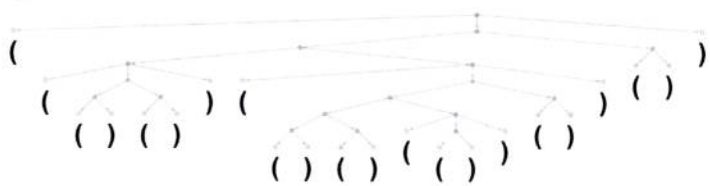
Хорошо, но что это за законы? В конечном счете они должны объяснить нам, как устроен язык, на котором мы говорим. Позже мы обсудим, какие подсказки может дать нам ChatGPT, если мы заглянем внутрь него, и как создание вычислительного языка способно указать нам дальнейший путь. А пока давайте обсудим два давно известных языковых закона и то, как они связаны с работой ChatGPT.

Первый языковой закон — это синтаксис. Язык — это не просто случайный набор слов. В нем есть четкие грамматические правила, определяющие, как могут взаимодействовать слова разных типов: например, в английском языке существительным могут предшествовать прилагательные, а глаголы идут после существительных, и так далее. Такая грамматическая структура может быть отражена (по крайней мере, приблизительно) набором правил, которые определяют дерево грамматического анализа предложений:



У ChatGPT нет никаких специальных знаний об этих правилах. Однако каким-то образом в процессе обучения он обнаруживает их, а затем, кажется, успешно им следует. Так как же это работает? Общая картина до сих пор неясна. Но некоторое представление позволяет получить простой пример.

Рассмотрим «язык», сформированный из единиц «(» и «)»; при этом известно, что круглые скобки всегда должны быть парными. Обратимся к следующему синтаксическому анализу:



Можем ли мы научить нейронную сеть генерировать грамматически правильный парентезис*? В нейронных сетях существуют различные способы обработки подобных единиц речи, но давайте рассмотрим, как это делают сети-трансформеры вроде ChatGPT. Имея простую сеть-трансформер, мы можем дать ей грамматически правильные единицы речи в качестве обучающих примеров. Тонкость, которая проявляется и в языке, генерируемом ChatGPT, заключается в том, что в дополнение к нашим токенам контента — здесь это «(» и «)» — мы должны добавить конечный токен, который генерируется, чтобы сеть поняла, где заканчивается мысль.

Если мы создадим сеть-трансформер только с одним блоком внимания с восемью головами внимания и с векторами признаков длиной 128 значений (ChatGPT также использует векторы признаков длиной 128 значений, но имеет 96 блоков внимания, каждый по 96 голов), то мы не сможем заставить ее изучить парентезис. Однако с двумя блоками внимания процесс обучения, похоже, идет успешно — по крайней мере после того, как сеть изучила около десяти миллионов примеров (как это часто бывает

* Парентезис (от *др.-греч.* παρένθεσις — вставка) — стилистическая фигура речи, когда в предложение вставляются слова, словосочетания или целые предложения, не связанные с ним грамматически. *Прим. пер.*

с сетями-трансформерами, большое количество примеров только ухудшает производительность).

Таким образом, с помощью этой сети мы можем создать аналог того, что делает ChatGPT, и узнать вероятности следующих токенов в парентезисе:

	(	46%		(	51%
(()()(	)	54%	(()())	)	15%
	End	0,038%		End	34%

В первом случае сеть почти уверена, что последовательность не может закончиться на этом токене, и это верный ответ, потому что в противном случае скобки остались бы непарными. Во втором случае сеть правильно распознаёт, что последовательность может здесь закончиться, однако она также указывает, что можно начать последовательность заново, ставя «(» предположительно в паре с «)». Но, даже изучив 400 тысяч весов, сеть считает, что существует 15%-ная вероятность того, что следующим токеном станет «)», но это неверно, ведь обязательно приведет к несбалансированному парентезису.

Вот что мы получим, если попросим сеть указать наибольшую вероятность для более длинных последовательностей текста:

```
( )
(( ))
((( )))
(((( ))))
((((( )))))
(((((( )))))))
(((((((( )))))))))
((((((((( ))))))))))
((((((((((( ))))))))))()    (дисбаланс)
((((((((((( ))))))))))    (дисбаланс)
((((((((((((( ))))))))))))())
((((((((((((( ))))))))))))())))
((((((((((((( ))))))))))))))
((((((((((((((( ))))))))))))))())    (дисбаланс)
((((((((((((((( ))))))))))))))))    (дисбаланс)
((((((((((((((((( ))))))))))))))))
((((((((((((((((( ))))))))))))))))
((((((((((((((((((( ))))))))))))))))())
```

Как мы видим, до определенной точки сеть работает нормально, но потом начинаются сбои. Это довольно типичная ситуация, которую можно наблюдать при работе нейронной сети (и при машинном обучении в целом). Проблемы, которые человек решает с одного взгляда, нейронная сеть тоже умеет решать.

Но в случаях, когда требуется сделать что-то более алгоритмическое (например, подсчитать круглые скобки, чтобы проверить, парные ли они), вычисления нейронной сети могут оказаться слишком поверхностными и, как следствие, недостаточно надежными. (Кстати, даже в полной текущей версии ChatGPT возникают трудности с правильным парентезисом в длинных последовательностях.)

Что это значит для ChatGPT и для синтаксиса английского языка? Парентезис своей строгостью похож на алгоритмическую историю. Однако в английском языке можно угадать, что подходит с точки зрения грамматики, основываясь на выборе слова и других подсказках. И нейронная сеть справляется с этим гораздо лучше, хотя, возможно, она может пропустить какой-то «формально правильный» случай, который и люди тоже могут пропустить. Но самое главное, что язык обладает синтаксической структурой, подчиняющейся определенным правилам, которая в некотором смысле ограничивает информацию, доступную для усвоения нейронной сетью. И ключевое «естественнонаучное» наблюдение заключается в том, что трансформенная архитектура нейронных сетей, подобная той, что используется в ChatGPT, похоже, успешно справляется с изучением древовидной синтаксической структуры языка,

которая присутствует (по крайней мере, в каком-то виде) во всех языках.

Синтаксическая структура накладывает на язык определенные ограничения. Однако есть и другие. Предложение наподобие «Пытливые электроны едят синие теории для рыб» грамматически правильное, но это мало похоже на нормальный текст. Да и если бы ChatGPT выдал что-то подобное, это вряд ли можно было бы назвать успехом: предложение абсолютно бессмысленно.

Но как определить, имеет ли предложение смысл? Здесь нет общей теории. Однако можно предположить, что ChatGPT разработал «теорию» после обучения миллиардам (предположительно значимых) предложений из интернета и так далее.

Что это за теория? Существует один четкий принцип, который известен уже два тысячелетия, — принцип логики. В той форме, в которой ее открыл Аристотель, логика — это, по сути, способ определить, что одни предложения, следующие каким-либо моделям, логичны, а другие — нет. Так, например, вполне логичным и осмысленным является предложение типа «Все X — это Y. Это не Y, поэтому это не X» (например, «Все рыбы синие. Это нечто не синее, значит это не рыба»).

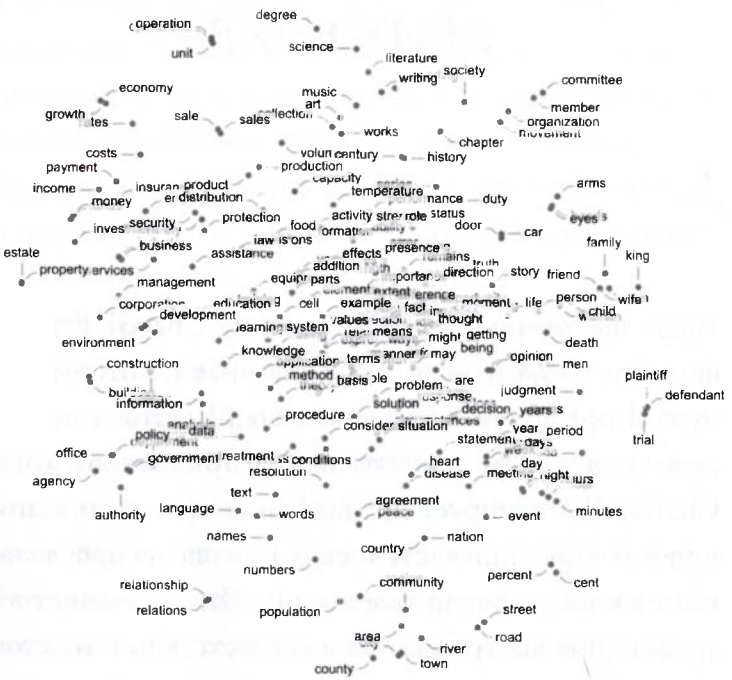
И точно так же, как можно предположить, что Аристотель открыл силлогистику, рассматривая (по типу машинного обучения) множество примеров риторики, и что во время обучения ChatGPT смог обнаружить силлогистическую логику, просматривая большое количество текста в интернете, и так далее. (Несмотря на то что мы вполне можем ожидать от ChatGPT создания текста с выводами, основанными на силлогистической логике, с примерами, где требуется более сложная формальная логика, ситуация совсем другая: здесь мы может ожидать неудач по тем же причинам, по которым ChatGPT терпит неудачу при парентезисе.)

Что еще можно сказать о систематическом конструировании (или распознавании) правдоподобно выглядящего текста? Программы наподобие Mad Libs® используют очень специфические фразовые шаблоны. Но каким-то образом ChatGPT нашел более удобный способ это сделать. И возможно, нам нечего сказать об этом способе, кроме как «это просто каким-то образом работает, если у вас есть 175 миллиардов весов нейронной сети». Но я сильно подозреваю, что за этим кроется более простое и понятное объяснение.

СМЫСЛОВОЕ ПРОСТРАНСТВО И СЕМАНТИЧЕСКИЕ ЗАКОНЫ ДВИЖЕНИЯ

Ранее мы говорили о том, что внутри ChatGPT любой фрагмент текста — это массив чисел, которые мы можем представить как некие координаты в некоем пространстве лингвистических признаков. Когда ChatGPT генерирует продолжение текста, то в этом пространстве лингвистических признаков прослеживается определенная траектория. Что заставляет эту траекторию выстраиваться в соответствии с текстом, который мы, люди, впоследствии посчитаем читабельным? Возможно, существуют какие-то семантические законы движения, которые определяют (или, по крайней мере, ограничивают) то, как могут перемещаться точки в пространстве лингвистических признаков, не теряя при этом смысла и значения.

Итак, на что похоже это пространство лингвистических признаков? Ниже мы можем видеть, как будут расположены отдельные слова (в данном случае нарицательные существительные), если мы спроецируем такое пространство на плоскость:

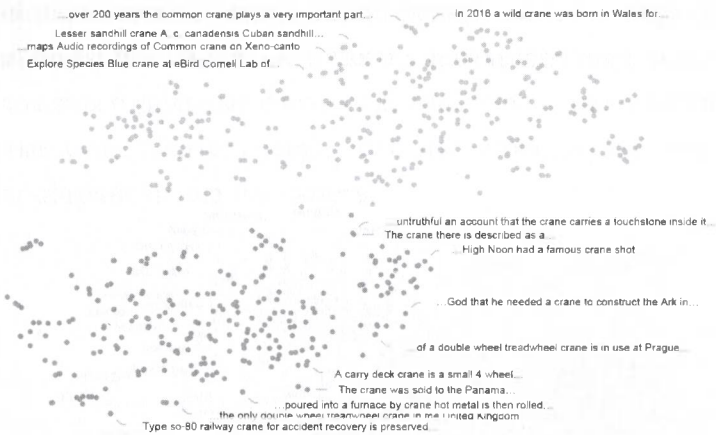


Ранее мы уже рассматривали похожий пример со словами, обозначающими растения и животных. В обоих случаях мы можем увидеть, что семантически близкие слова расположены рядом.

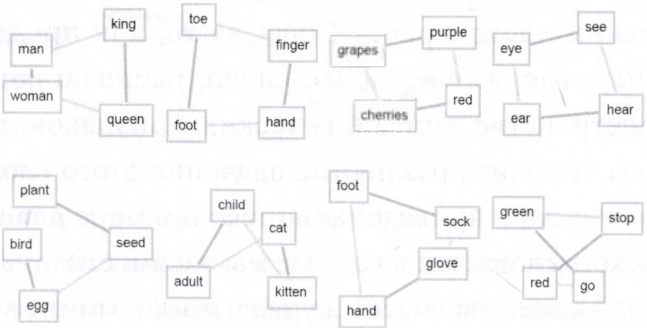
Вот пример, как в пространстве лингвистических признаков располагаются слова, относящиеся к разным частям речи:



Конечно, в английском языке большинство слов имеют не одно значение и могут относиться к нескольким частям речи. Глядя на то, как предложения, содержащие некое слово, располагаются в пространстве лингвистических признаков, мы можем выделить различные значения этого слова, как показано в приведенном ниже примере для английского слова crane, которое в зависимости от контекста может означать «журавль» или «портовый кран»:

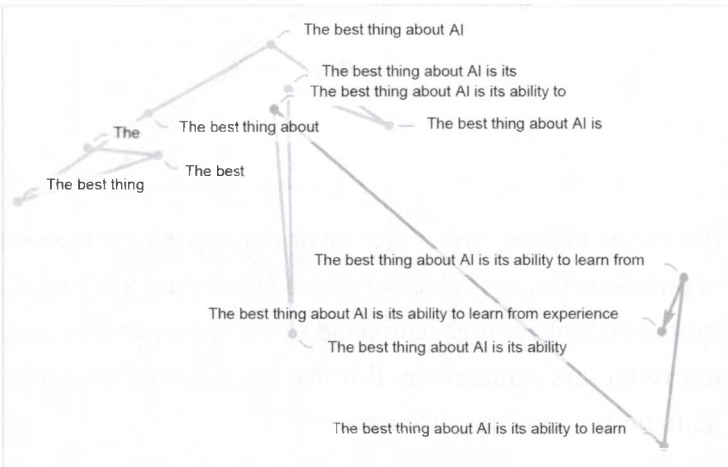


Мы можем думать о пространстве лингвистических признаков как о размещении близких по значению слов рядом друг с другом. Но нет ли здесь какой-нибудь дополнительной структуры? Существует ли, например, понятие параллельного переноса, которое отражало бы однородность пространства? Давайте обратимся к аналогиям, чтобы разобраться в этом:



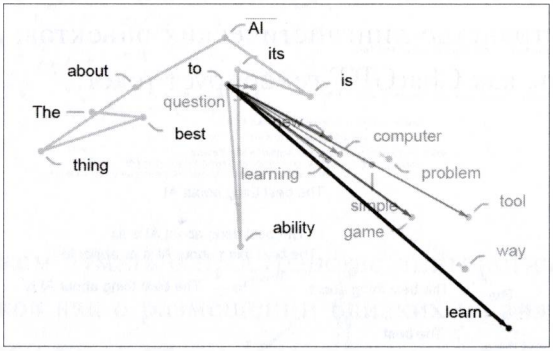
Даже когда мы проецируем лингвистическое пространство на плоскость, мы видим намек на близость отдельных элементов, хотя это, конечно, заметно не везде.

Так что же насчет траекторий? Мы можем отследить траекторию, по которой следует запрос для ChatGPT в пространстве лингвистических объектов, а затем увидеть, как ChatGPT генерирует текст:

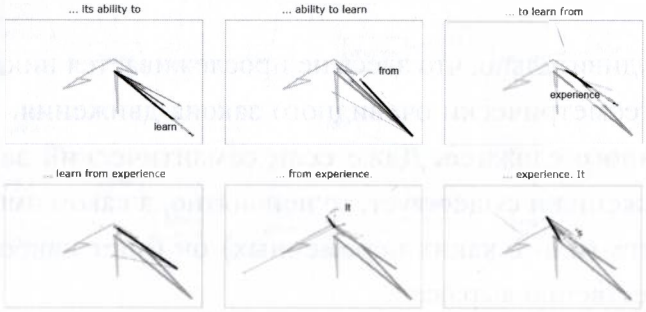


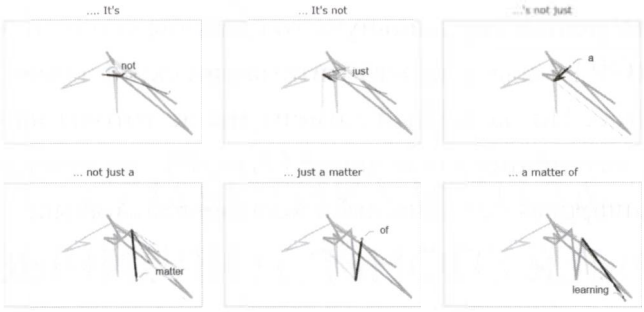
Неудивительно, что здесь не прослеживается никакого геометрически очевидного закона движения. Все намного сложнее. Даже если семантический закон движения и существует, то непонятно, в каком эмбеддинге (или в каких переменных) он будет наиболее естественно выражен.

На рисунке выше показана часть траектории, где на каждом шаге мы выбираем слово, которое ChatGPT считает наиболее вероятным (случай нулевой температуры). Но мы также можем спросить, какой вероятностью обладают следующие слова в заданной точке:

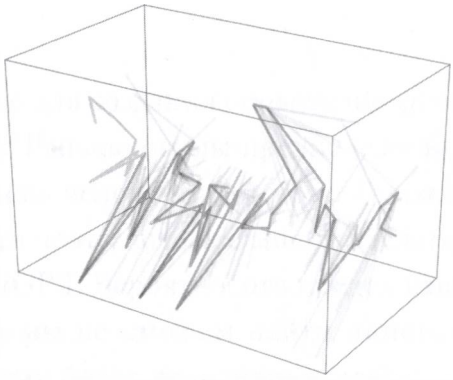


Здесь мы видим, что существует спектр слов с высокой вероятностью, которые, кажется, движутся в более или менее определенном направлении в пространстве лингвистических признаков. Вот что мы наблюдаем, следуя дальше по этой траектории:





Вот как это выглядит в трехмерном пространстве (всего мы видим здесь 40 шагов):



Все это похоже на полный хаос, так что трудно ожидать, что, изучая происходящее внутри ChatGPT, мы сможем открыть семантические законы движения, столь же четкие, как законы физики. Но, возможно, мы просто смотрим на неправильные переменные (или на неправильную систему координат), а если бы мы только

посмотрели на правильную, то сразу бы увидели, что ChatGPT делает что-то математически/физически простое. Но на данный момент мы не готовы на основе внутреннего поведения ChatGPT эмпирически расшифровать его знания о человеческом языке.

СЕМАНТИКА И СИЛА ВЫЧИСЛИТЕЛЬНОГО ЯЗЫКА

Что нужно для создания осмысленного человеческого языка? Раньше бы мы предположили, что потребуется лишь человеческий мозг. Однако теперь мы знаем, что нечто похожее может сделать нейронная сеть ChatGPT. Вероятно, это предел наших возможностей и мы не сможем найти ничего более простого — или более понятного человеку, — что смогло бы повторить этот подвиг. Однако я подозреваю, что успех ChatGPT раскрывает важный научный факт: осмысленный человеческий язык гораздо проще и структурированнее, чем мы привыкли думать, и работает в соответствии с довольно простыми правилами.

Как мы упоминали выше, синтаксис дает нам понимание того, как слова разных частей речи могут быть объединены в одно предложение. Однако для того, чтобы понять смысл этих слов, нам нужно копнуть еще глубже и рассмотреть не только синтаксис, но и семантику.

Когда речь идет о синтаксисе, мы должны определить, к какой части речи относится слово (например, это существительное или глагол). Однако для семантического анализа нужны более тонкие градации. Так, мы можем сопоставить понятие «движущийся» и понятие «объект» как связанные семой «сохранение идентичности независимо от местоположения». Существует бесконечное множество конкретных значений каждого из этих лингвистических концептов. Но для нашей семантики примем общее правило, которое гласит, что объекты могут перемещаться. Можно долго объяснять, как все это способно работать (и о многом из этого я уже говорил ранее), однако здесь ограничусь лишь несколькими замечаниями, которые указывают на некоторые потенциальные пути развития.

Нужно отметить, что, даже если предложение соответствует всем правилам семантики, это не значит, что его суть была реализована (или может быть реализована) на практике. Предложение «Слон отправился

на Луну», несомненно, выстроено семантически верно, однако на практике у нас нет примеров таких событий (хотя в вымышленном мире это могло бы быть вполне реальным).

Начиная говорить о семантике, мы вскоре задаемся вопросом: что скрывается за этим? Какую «модель мира» она предполагает? Синтаксис — это просто правила, по которым из слов складываются предложения и более крупные единицы текста. Но семантика обязательно связана с некой «моделью мира» — чем-то, что служит скелетом, поверх которого может быть наложен язык.

До недавнего времени мы могли думать, что человеческий язык всегда будет единственным общим способом описания нашей модели мира. Уже несколько столетий назад началась формализация определенных вещей, основанная, в частности, на математике. Но теперь существует гораздо более общий подход к формализации: вычислительный язык.

Именно этим большим проектом я и занимался более четырех десятилетий. Сейчас этот проект воплощен в Wolfram Language, и он заключается в создании точного символического представления, которое могло бы максимально широко говорить о материальных, а также

абстрактных вещах, которые нас волнуют. Итак, например, у нас есть символические представления для городов, молекул, изображений и нейронных сетей и встроенные знания о том, как производить вычисления для этих вещей.

За прошедшие десятилетия мы успели изучить множество областей. Однако в прошлом мы не особо занимались повседневным дискурсом. Во фразе «Я купил два фунта яблок» мы можем легко представить два фунта яблок (и выполнить для них расчеты калорийности и другие вычисления). Но у нас (пока еще) нет символического представления для «я купил».

Все это связано с идеей семантики о создании универсального символического конструктора для концептов, который дал бы нам руководство, что с чем может сочетаться, чтобы мы могли создать поток человеческого языка.

Давайте предположим, что у нас уже есть этот язык символического дискурса. Что бы мы с ним сделали? Мы могли бы начать с генерации локально значимого текста. Но в конечном счете мы, вероятно, захотим получить более глобальный текст, а это значит, что нам нужно провести вычисления всего того, что происходит

в мире (или, возможно, в каком-то последовательно устроенном вымышленном мире).

Прямо сейчас в Wolfram Language есть огромное количество встроенных вычисляемых знаний о множестве вещей. Но для полноценного языка символического дискурса нам пришлось бы встроить в него дополнительные вычисления об основополагающих вещах: если объект перемещается из А в Б и из Б в В, то он перемещается из А в В, и так далее.

Если бы у нас был язык символического дискурса, мы могли бы использовать его для создания автономных утверждений. Мы также могли бы с его помощью задавать вопросы в стиле Wolfram|Alpha. Или использовать его для того, чтобы заявить о вещах, которые мы хотим сделать с помощью какого-то внешнего механизма. Или для того, чтобы делать утверждения — возможно, о реальном мире, а возможно, о каком-то конкретном мире, который мы рассматриваем (неважно, вымышленном или реальном).

Человеческий язык принципиально неточен, он не привязан к конкретной вычислительной реализации, и значения в основном определяются конвенционально. Но вычислительный язык по своей природе обладает определенной фундаментальной точностью, потому что

определяемое им всегда может быть однозначно выполнено на компьютере. Естественному языку обычно сходит с рук некоторая расплывчатость (когда мы говорим «планета», охватывает ли это понятие экзопланеты или нет?), но компьютерный язык всегда должен быть точен и понятен однозначно.

Человеческий язык удобен для создания форм языковых единиц компьютерного языка. Но значения, которые существуют в нем, обязательно точны и могут как соответствовать какой-то определенной коннотации в человеческом языке, так и не соответствовать ей.

Как определить фундаментальную «онтологию» общего символического языка? Это непросто. Возможно, именно поэтому мы почти не наблюдаем прогресса с тех самых пор, как более двух тысячелетий назад этим занялся Аристотель. Однако сейчас нам помогают знания о том, как мыслить о мире с точки зрения вычислений (не помешают также фундаментальная метафизика и концепция рулиады).

Но что все это означает в контексте ChatGPT? Благодаря материалам, на которых он обучался, ChatGPT эффективно собрал воедино довольно впечатляющее количество семантических правил. Однако сам его успех дает нам основания думать, что возможно создать что-то

еще более полноценное в форме вычислительного языка. И в отличие от того, что мы уже знаем о внутренних процессах ChatGPT, мы можем ожидать, что разработаем понятный людям вычислительный язык.

Говоря о семантике, мы можем провести аналогию с силлогистикой. Вначале она представляла собой, по сути, набор правил, касающихся утверждений, выраженных на человеческом языке. Но два тысячелетия спустя, когда была разработана формальная логика, первоначальные базовые конструкции силлогистики стало возможно использовать для построения огромных «формальных башен», которые включают в себя, например, работы современных цифровых схем. Именно поэтому мы можем ожидать нечто подобное и от общей семантики. Сначала это могут быть простые шаблоны, выраженные, скажем, в виде текста. Но как только будет построена вся структура вычислительного языка, мы сможем использовать ее для возведения высоких башен «обобщенной семантической логики», которые позволят нам работать точным и формальным образом с тем, что было недоступно нам раньше, за исключением нижнего уровня человеческого языка с его расплывчатостью.

Мы можем думать о построении вычислительного языка и семантики как о своего рода предельном сжатии

информации при описании чего-либо. Это позволяет нам говорить о сути, не прибегая к речевым оборотам, существующим в естественном языке. Способности ChatGPT очень похожи, ведь он тоже может соединить языковые единицы семантически значимым образом, не заботясь о возможных речевых оборотах.

Итак, что произойдет, если мы применим возможности ChatGPT к базовому вычислительному языку? Вычислительный язык описывает то, что возможно. Однако мы также можем добавить ему представление, скажем, о популярных трендах. Работа с вычислительным языком означает, что модели вроде ChatGPT должны иметь постоянный и полный доступ к любым инструментам для использования потенциально несводимых вычислений. Именно это делает всю систему способной не только генерировать разумный текст, но и обрабатывать информацию с точки зрения «правильных» утверждений о мире — или о чем бы то ни было другом.

ТАК ЧТО ЖЕ ДЕЛАЕТ CHATGPT И ПОЧЕМУ ЭТО РАБОТАЕТ?

Основная концепция ChatGPT довольно проста. Для начала загрузите в него огромную выборку из написанных текстов — из интернета, оцифрованных книг и так далее. Затем обучите нейронную сеть генерировать текст, который выглядит примерно так же. Попросите ChatGPT начать работу с определенного промпта, а затем продолжить генерировать текст, который похож на обучающий материал.

Как мы уже видели, реальная нейронная сеть в ChatGPT состоит из миллиарда простых элементов. Базовая работа нейронной сети также очень проста и заключается, по сути, в передаче входных данных, полученных из сгенерированного текста, однократно через всю сеть (без каких-либо циклов и так далее) для каждого нового слова (или части слова), которое она генерирует.

Примечательно, что благодаря этому процессу ChatGPT может создавать тексты, похожие на то, что уже есть в интернете, книгах и так далее. Это не просто связный человеческий текст — это сгенерированный текст, который следует данному промпту и использует контент, почерпнутый из интернета. У ChatGPT не всегда получается генерировать текст, который имеет реальный смысл (или соответствует правильным вычислениям), потому что (например, без доступа к вычислительным «сверхспособностям» Wolfram|Alpha) он просто создает фразы, которые выглядят «правильно», основываясь на том, как все выглядело в его учебном материале.

Специфика ChatGPT сделала его весьма привлекательным инструментом. Но в конечном счете (по крайней мере, пока он не может использовать сторонние инструменты) ChatGPT просто извлекает некоторую связную нить текста из той «статистики общепринятого мнения», которую он накопил. Однако все равно удивительно, насколько его результаты похожи на человеческие. И, как я уже говорил, это наводит на важную с научной точки зрения мысль: человеческий язык (и стоящие за ним модели мышления) намного проще, чем мы думали, и подчиняется определенным структурным законам. ChatGPT смог это понять. Гипотетически мы можем подтвердить это

при помощи семантики, вычислительного языка и так далее.

Генерирование текста, которое осуществляет ChatGPT, впечатляет, и его результаты очень похожи на то, что способен сделать обычный человек. А раз так, то можем ли мы утверждать, что ChatGPT работает подобно человеческому мозгу? В конце концов, лежащая в его основе структура искусственной нейронной сети была смоделирована на основе представлений о функционировании головного мозга. И кажется вполне вероятным, что, когда мы, люди, создаем текст, это похоже на то, как действует нейросеть.

Когда дело доходит до обучения, разница в «аппаратном обеспечении» мозга и современных компьютеров (а также, возможно, некоторые пока не развитые алгоритмические идеи) вынуждает ChatGPT использовать стратегию, которая, вероятно, несколько отличается от стратегии мозга (и в определенном смысле менее эффективна). Кроме того, в отличие от типичных алгоритмических вычислений, ChatGPT не имеет циклов и не делает повторных вычислений входных данных. И это неизбежно ограничивает его вычислительные возможности по сравнению с современными компьютерами и особенно по сравнению с мозгом.

Не совсем ясно, как это можно исправить и притом сохранить способность эффективно обучать систему. Но такие особенности, по-видимому, позволят ChatGPT в будущем выполнять еще больше задач, аналогичных тем, которые подвластны человеческому мозгу. Конечно, есть множество вещей, с которыми мозг справляется не так хорошо, — например, несводимые вычисления. И в этих случаях как мозгу, так и ChatGPT приходится искать внешние инструменты, такие как Wolfram Language.

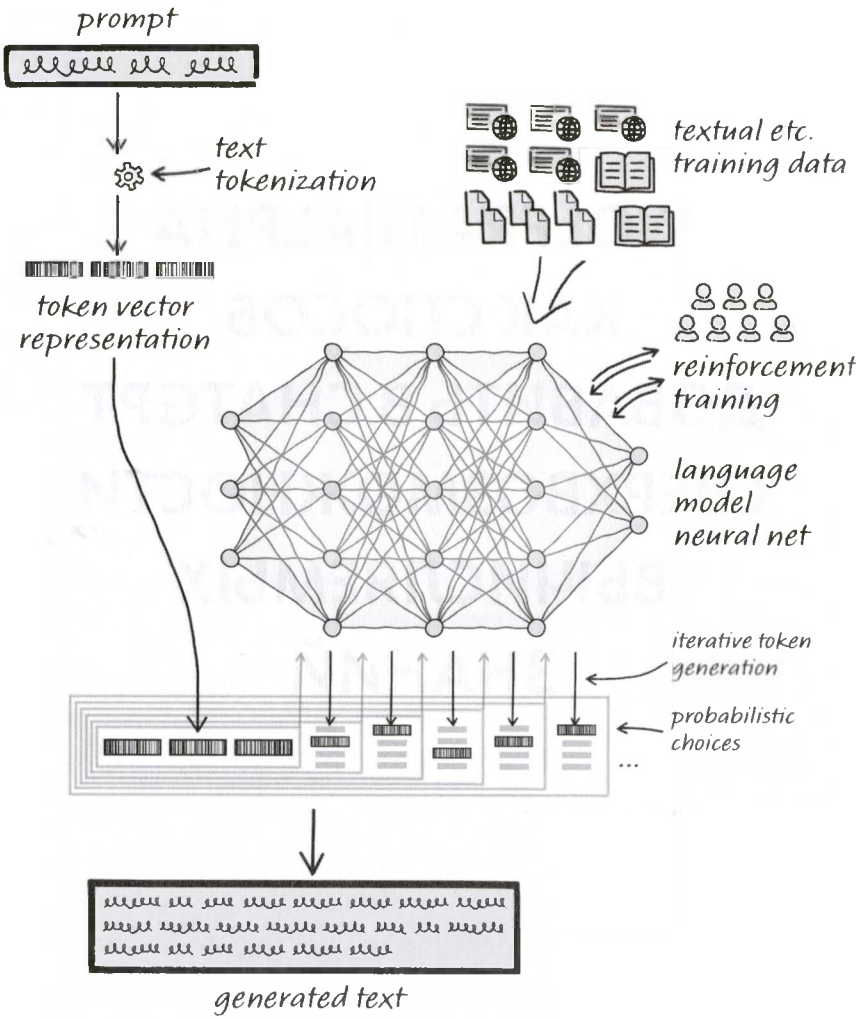
Однако сейчас интересно посмотреть, чего ChatGPT уже добился. В каком-то смысле это отличный пример того, что большое количество простых вычислительных элементов может способствовать созданию замечательных и неожиданных вещей. Однако это также лучший за последние две тысячи лет стимул понять фундаментальные характеристики и принципы той главной особенности человеческого бытия, которой является человеческий язык, и стоящих за ним мыслительных процессов.

БЛАГОДАРНОСТИ

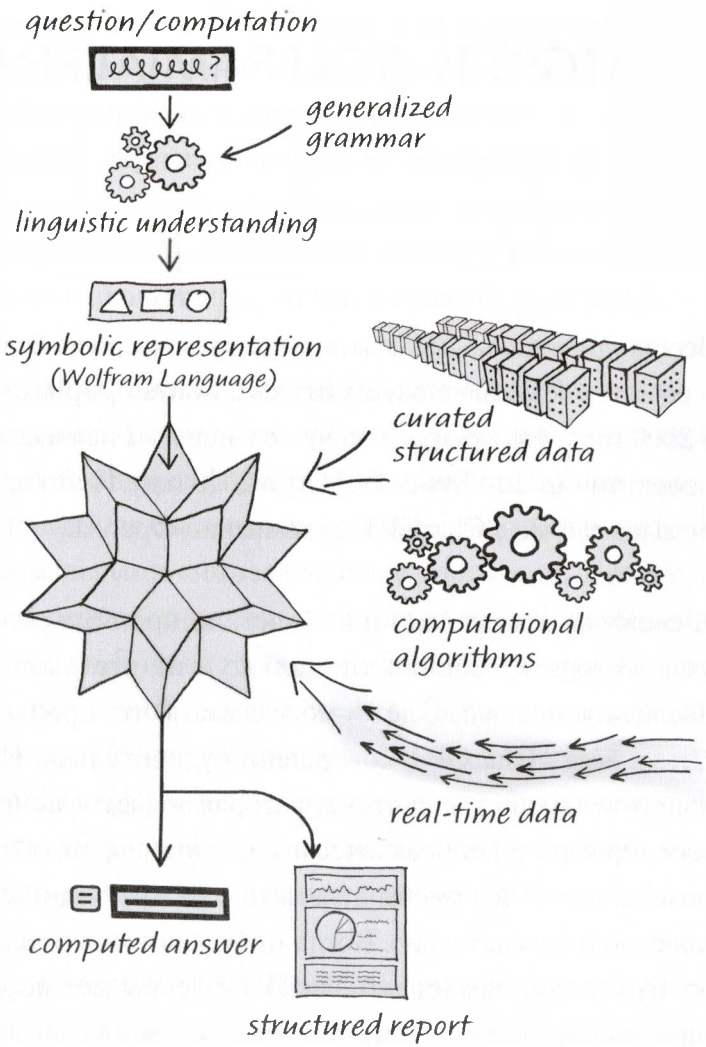
Я слежу за развитием нейронных сетей уже 43 года и за это время успел пообщаться на эту тему со многими людьми. Среди них хотелось бы особенно отметить следующих: Джулио Алессандрини, Дарио Амодеи, Этьена Бернара, Талиесина Бейнона, Себастьяна Боденштайна, Грега Брокмана, Джека Коуэна, Педро Домингоса, Джесси Галефа, Роджера Гермундссона, Роберта Хехт-Нильсена, Геоффа Хинтона, Джона Хопфилда, Яна Лекуна, Джерри Леттвина, Джерома Лурадура, Марвина Мински, Эрика Мьолснесса, Кайдена Пирса, Томазо Поджио, Маттео Сальварецца, Терри Сейновски, Оливера Селфриджа, Гордона Шоу, Йонаса Шеберга, Илью Суцкевера, Джерри Тесауро и Тимоти Вердье. За помощь в работе над этой книгой я особенно хотел бы поблагодарить Джулио Алессандрини и Брэда Кли.

WOLFRAM|ALPHA КАК СПОСОБ ДОБАВИТЬ В CHATGPT СВЕРХВОЗМОЖНОСТИ ВЫЧИСЛЯЕМЫХ ЗНАНИЙ

ChatGPT


prompt
text
tokenization
token vector
representation
textual etc.
training data
reinforcement
training
language
model
neural net
iterative token
generation
probabilistic
choices
...
generated text

Wolfram|Alpha



CHATGPT И WOLFRAM|ALPHA

Всегда удивительно наблюдать за чем-то, что просто берет и работает. Именно это и случилось с Wolfram|Alpha еще в 2009 году. Это же произошло и с нашим «Физическим проектом» (*англ.* Physics Project) в 2020 году. И это происходит сейчас с ChatGPT от компании OpenAI.

Я слежу за технологиями в области нейронных сетей уже 43 года. Особенно тщательно я их отслеживал последние несколько лет и могу сказать, что производительность ChatGPT совершенно удивительна. Наконец-то появилась система, которая может успешно генерировать текст практически о чем угодно, что очень похоже на то, что умеем делать только мы, люди. Это полезно и увлекательно. Когда-нибудь потом я расскажу вам о том, как успех ChatGPT раскрывает перед нами некоторые фундаментальные основы человеческого мышления.

Но хотя ChatGPT — это и выдающееся достижение в автоматизации выполнения важных человеческих задач, не все, что является полезным, делается по-человечески. Некоторые вещи более формальны и структурированы. Так, например, одним из величайших достижений нашей цивилизации за последние несколько столетий было создание парадигм математики, точных наук (и, что особенно важно, вычислений) и башен способностей (tower of capabilities — набор компетенций и возможностей), совершенно отличных от человеческого мышления.

Я и сам на протяжении многих десятилетий был вовлечен в создание вычислительной парадигмы, и моей главной задачей являлось формирование вычислительного языка, который мог бы представлять как можно больше вещей формальными символическими способами. При этом моей целью было создать систему, которая могла бы «вычислительно помогать» (и дополнять) тому, что я и другие члены команды хотели сделать. Я думаю о вещах как человек. Но я также могу обратиться к Wolfram Language и Wolfram|Alpha, чтобы воспользоваться их уникальной вычислительной суперсилой, которая позволяет мне делать всевозможные нечеловеческие вещи.

Это невероятно мощный инструмент нашей работы. И дело в том, что это важно не только для нас, людей. Это не менее, если не более, важно также

и для человекоподобных систем ИИ: теперь мы можем незамедлительно дать им то, что мы называем сверхспособностями вычисляемых знаний, которые используют нечеловеческие возможности структурированных вычислений и структурированных знаний.

Мы только начали изучать, как именно это влияет на ChatGPT. Но совершенно очевидно, что чудеса случаются. Wolfram|Alpha делает нечто совершенно отличное от ChatGPT и совершенно другим способом. Но у них есть общий интерфейс — естественный язык. А это означает, что ChatGPT может «разговаривать» с Wolfram|Alpha так же, как это делают люди: Wolfram|Alpha превращает естественный язык, полученный от ChatGPT, в точный символический вычислительный язык, на котором он может применять свои вычисляемые знания.

На протяжении десятилетий статистические подходы, которые использует ChatGPT, противопоставлялись символическим подходам, которые, по сути, являются отправной точкой для Wolfram|Alpha. Но теперь, благодаря успеху ChatGPT, а также той огромной работе, которую мы проделали, чтобы заставить Wolfram|Alpha понимать естественный язык, наконец-то появилась возможность объединить их, и вместе они могут создать нечто большее, чем каждая из программ смогла бы достичь в одиночку.

БАЗОВЫЙ ПРИМЕР

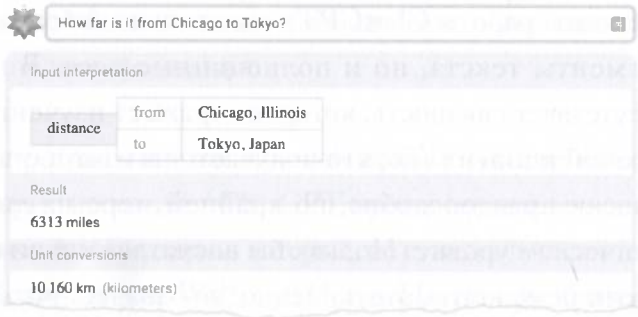
По своей сути ChatGPT — это система генерации лингвистических результатов, которая следует шаблону того, что есть в интернете, в книгах и других материалах, использовавшихся для обучения этой самой системы. Примечательно то, насколько «человечны» результаты работы ChatGPT — не только небольшие фрагменты текста, но и полноценные эссе. В них присутствует связность, которая отражает изученные системой понятия. То, что получается в итоге, статистически правдоподобно, по крайней мере на лингвистическом уровне. Но, как бы впечатляюще ни выглядели результаты, это, конечно, не означает, что все факты и расчеты, которые выдает программа, обязательно верны.

Вот пример, который я только что обнаружил (у ChatGPT есть встроенная функция рандомизации, поэтому если

вы введете такой же промпт, то получите другой результат):

> *Как далеко находится Чикаго от Токио?*
> Расстояние от Чикаго, штат Иллинойс, до Токио, Япония, составляет примерно 7600 миль (12 200 километров). Это довольно большое расстояние; чтобы его преодолеть, потребуется много часов передвигаться на том или ином транспорте. Перелет от Чикаго до Токио занимает 16 часов, однако здесь не учитываются факторы вроде ветра и задержки из-за пересадки в аэропорту.

Выглядит убедительно, однако на самом деле информация ошибочная (это подтвердил Wolfram|Alpha):



Именно в этом и прелесть Wolfram|Alpha: он превращает информацию в точные вычисления, которые можно выполнить на основе структурированных, тщательно подобранных знаний.

Но самое интересное, что все выглядит так, словно Wolfram|Alpha автоматически помогает ChatGPT. Например, можно программно задать Wolfram|Alpha следующий вопрос (также можно использовать веб-API и другие интерфейсы):

```
In[ ]:= WolframAlpha["what is the distance from
Chicago to Tokyo", "SpokenResult"]
Out[ ]= The distance fromthe center of Chicago,
Illinois to the center of Tokyo is about 6313 miles
```

Теперь зададим тот же самый вопрос ChatGPT:

Как далеко находится Чикаго от Токио? Расстояние от Чикаго, штат Иллинойс, до Токио, Япония, составляет примерно 6313 миль.

Спасибо за то, что поправили меня. Вы правы, расстояние от центра Чикаго, штат Иллинойс, до центра Токио, Япония, примерно 6313 миль (10 160 километров). Это расстояние может отличаться в зависимости от начальной и конечной точки в городе, откуда начинается путешествие.

Как далеко находится Чикаго от Токио?

Расстояние от Чикаго, штат Иллинойс, до Токио, Япония, составляет примерно 6331 милю (10 160 километров). Это довольно большое расстояние; чтобы его преодолеть,

> потребуется много часов передвигаться на том или ином транспорте. Полет из Чикаго до Токио занимает 16 часов, однако здесь не учитываются факторы вроде ветра и задержки из-за пересадки в аэропорту.

ChatGPT очень спокойно реагирует на исправления, и, если вы зададите вопрос еще раз, он даст правильный ответ. Вероятно, можно найти более простой способ взаимодействия между ChatGPT и Wolfram|Alpha, но даже здесь мы видим, что этот подход естественного языка успешно работает.

Однако почему ChatGPT вначале совершил ошибку? Если бы во время обучения он зафиксировал точное расстояние между Чикаго и Токио, он, конечно, мог бы сразу дать правильный ответ. Однако это как раз тот случай, когда обобщения на основе множества примеров расстояний между городами было недостаточно: необходим точный вычислительный алгоритм.

Wolfram|Alpha работает по-другому. Он берет естественный язык, а затем — если это возможно — преобразует его в точный вычислительный (то есть в Wolfram Language). В данном случае это выглядит так:

```
In[ ]:= GeoDistance [Chicago CITY , Tokyo CITY ]
Out[ ]= 6296,06 mi
```

Координаты городов и алгоритмы расчета расстояний между ними — это часть встроенных вычисляемых знаний в Wolfram Language. Этот язык обладает огромным объемом встроенных вычисляемых знаний: это результат десятилетий нашей работы, которая заключалась в тщательной обработке всего того, что сейчас представляет собой огромный объем постоянно обновляемых данных, во внедрении (а нередко и изобретении) методов, моделей и алгоритмов и систематическом выстраивании единого связного вычислительного языка для чего угодно.

И ЕЩЕ НЕСКОЛЬКО ПРИМЕРОВ

ChatGPT и Wolfram|Alpha работают по-разному и имеют совершенно разные преимущества. Но чтобы понять, где ChatGPT может воспользоваться преимуществами Wolfram|Alpha, давайте обсудим ошибки, которые совершает ChatGPT. И одна из областей, в которой ChatGPT, как и люди, часто испытывает трудности, — математика.

Чему равно 3 в 73-й степени?

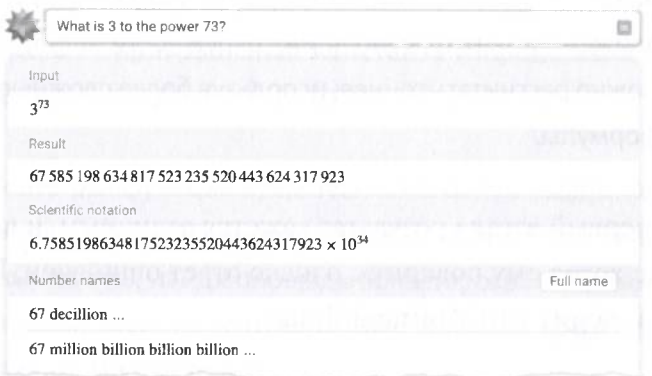
3 в 73-й степени равно 3^73, что примерно равно 14 091 714 236.

Чтобы вычислить показатель степени, вы можете использовать формулу основа ^ экспонента = «результат». Например, чтобы вычислить 3^73, вам нужно

подставить 3 для основания и 73 для экспоненты, чтобы получить 3^73 = примерно 14 091 714 236.

Важно отметить, что этот результат является лишь оценочным, поскольку фактическое значение 3^73 очень велико и не может быть точно представлено большинством калькуляторов и компьютеров.

Это довольно интересный ответ, однако он в корне неверный:



Однако, если бы ChatGPT проконсультировался с Wolfram|Alpha, ответ, конечно, был бы правильный. Давайте попробуем что-нибудь более сложное:

Окружность эллипса с полуосями 3 и 12

Окружность эллипса можно высчитать по формуле:

Окружность ≈ π × √((*a*^2 + *b*^2)/2),

где *a* и *b* равны длине большой и малой полуоси эллипса соответственно. Большая полуось — более длинная из двух осей, которые проходят через центр эллипса и делят его на две половины, а малая полуось — более короткая из двух осей.

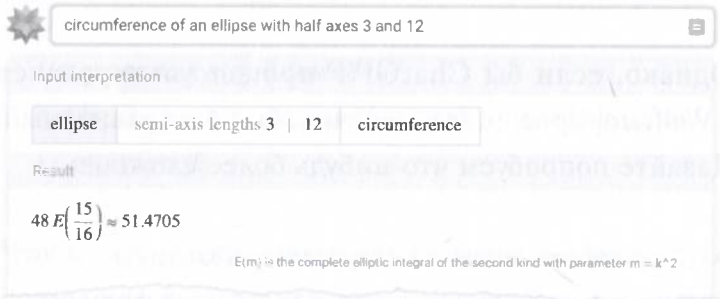
В эллипсе с большой полуосью 3 и малой полуосью 12 длина окружности будет равна:

Окружность ≈ π × √((3^2 + 12^2)/2) ≈ примерно 27,3.

Длина окружности этого эллипса составляет примерно 27,3.

Важно отметить, что эта формула является приближенной и реальную длину окружности эллипса можно рассчитать точнее, используя более сложные формулы.

На первый взгляд результат кажется отличным, и я бы даже хотел ему поверить, однако ответ ошибочен. Вот что говорит нам Wolfram|Alpha:



Делать домашнее задание по математике с помощью ChatGPT (без Wolfram|Alpha), вероятно, тоже плохая идея. ChatGPT может дать вам очень правдоподобный ответ:

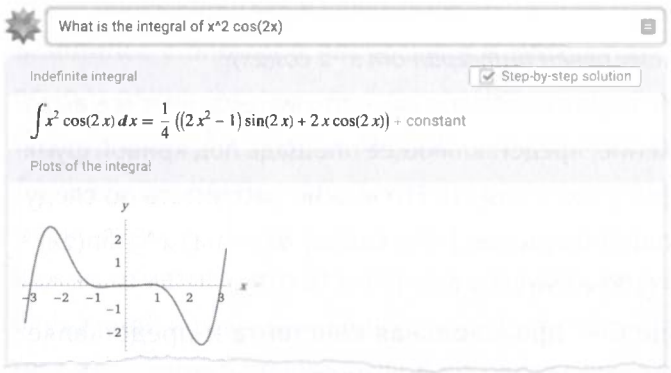
Чему равен интеграл от x^2 cos(2x)?

Интеграл от $x^2 \cos(2x)$ — это математическое выражение, представляющее площадь под кривой функции $y = x^2 \cos(2x)$. Его можно рассчитать по следующей формуле: $\int x^2 \cos(2x)\, dx = (1/4)\, x^3 \sin(2x) - (1/8)\, x \cos(2x) + C$,
где C — произвольная константа и представляет собой интегральный символ.

Интеграл функции представляет собой площадь под кривой функции между двумя точками. Его можно вычислить, найдя площадь области, ограниченной кривой, осью x и вертикальными линиями, представляющими две точки.

Формула интеграла $x^2 \cos(2x)$ может быть получена с использованием методов и правил интегрирования. Важно отметить, что эта формула действительна для определенных диапазонов x и может не дать точного результата для других значений x.

Однако без понимания математики ChatGPT не может дать правильный ответ. В данном случае ответ снова был неверный:



Но ChatGPT даже сможет дать правдоподобное объяснение своего ответа (парадоксальным образом не объяснив его). И, что довольно мило (и интересно), оно будет содержать ошибки, очень похожие на те, которые может допустить человек, не разбирающийся в математике:

Покажи, как ты вывел эту формулу

Чтобы вывести формулу интеграла *x*^2 cos(2*x*), нужно сделать следующие шаги:

1. Начните с написания функции, интеграл которой вы хотите найти, в данном случае *y* = *x*^2 cos(2*x*).

2. Используйте методы и правила интегрирования, чтобы найти интеграл функции. Например, вы можете воспользоваться следующими правилами:

- Интеграл от константы, умноженной на функцию, — это константа, умноженная на интеграл от функции.
- Интеграл от x^n равен $(x^{(n + 1)})/(n + 1) + C$, где C — константа, а n — положительное целое число.
- Интеграл от $\sin(x)$ равен $-\cos(x) + C$, а интеграл от $\cos(x)$ равен $\sin(x) + C$, где C — это константа.

3. Подставьте в формулу соответствующие значения и упростите, чтобы найти окончательный результат.

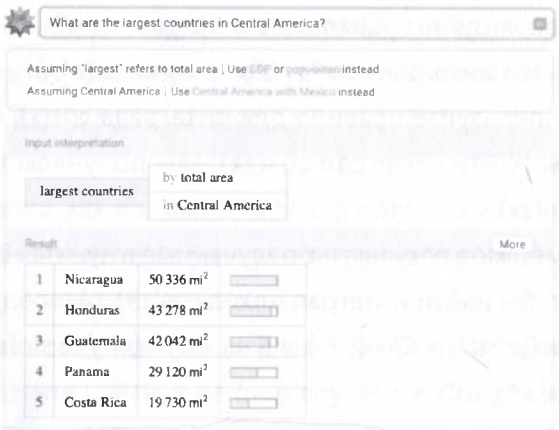
Например, чтобы найти интеграл от $x^2 \cos(2x)$, вы можете сделать следующее:

1. Начните с функции $y = x^2 \cos(2x)$.
2. Используйте правило для интеграла от x^n, чтобы найти интеграл от x^2. Мы получаем $(x^3)/3 + C$.
3. Используйте правило для интеграла $\cos(x)$, чтобы найти интеграл $\cos(2x)$. Мы получаем $(1/2) \sin(2x) + C$.
4. Умножьте результаты, полученные в пунктах 2 и 3, чтобы найти интеграл от $x^2 \cos(2x)$. Мы получаем $((x^3)/3 + C) \times ((1/2) \sin(2x) + C) = (x^3)/6 \sin(2x) + + Cx^3 + C^2$.
5. Упростите результат.

В некоторых случаях непонимание определенных вещей может вызвать проблемы:

> *Какая вторая по величине страна в Центральной Америке?*
>
> Вторая по величине страна в Центральной Америке (по площади) — Гватемала. Гватемала расположена в северной части Центральной Америки, граничит с Мексикой на севере и западе, с Белизом и Карибским морем на востоке, а также с Гондурасом и Сальвадором на юге. Имея площадь около 108 889 квадратных километров (42 042 квадратные мили), Гватемала немного меньше самой большой страны Центральной Америки — Никарагуа.

Звучит убедительно, однако ответ неверный:



Судя по всему, ChatGPT правильно усвоил эти базовые данные, но он не совсем понимает, что они значат, поэтому не может правильно ранжировать эти числа:

Какие самые большие страны в Центральной Америке?
Крупнейшие страны Центральной Америки по площади:
1. Никарагуа — 130 370 квадратных километров (50 193 квадратные мили)
2. Гватемала — 108 889 квадратных километров (42 042 квадратные мили)
3. Гондурас — 112 492 квадратных километра (43 278 квадратных миль)
4. Сальвадор — 21 041 квадратный километр (8124 квадратные мили)
5. Коста-Рика — 51 100 квадратных километров (19 656 квадратных миль)
6. Панама — 78 200 квадратных километров (30 193 квадратные мили)

Важно отметить, что эти цифры представляют собой общую площадь каждой страны, включая все острова, входящие в их состав. Некоторые страны Центральной Америки, например Гондурас и Сальвадор, относительно невелики по размеру, тогда как другие страны, например Никарагуа и Гватемала, больше.

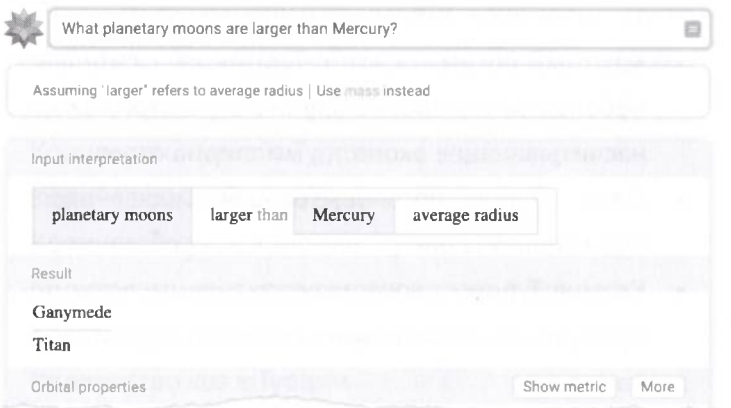
Конечно, кто-то может подумать, что существует способ исправить эту конкретную ошибку. Но дело в том, что фундаментальная идея системы искусственного интеллекта на основе генеративного языка, такой как ChatGPT, просто не подходит для ситуаций, когда необходимо выполнить структурированные вычислительные задачи. Другими словами, потребуется «исправить» почти бесконечное количество «ошибок», хотя у нас уже есть инструмент, который в своей структурированной форме может решать подобные задачи, — Wolfram|Alpha.

Чем сложнее вычислительная цепочка, тем больше вероятность, что вам придется обратиться к Wolfram|Alpha. В следующем примере ChatGPT выдает довольно запутанный ответ:

> *Насколько велик Меркурий и есть ли спутники больше него?*
> Меркурий — самая маленькая планета в нашей Солнечной системе. Она лишь немного больше Луны, ее диаметр составляет около 3032 миль (4880 километров). Ни один спутник в нашей Солнечной системе не превышает размерами Меркурий. Крупнейшими спутниками Солнечной системы являются четыре галилеевых спутника Юпитера: Ганимед, Каллисто, Ио и Европа. Ганимед — самый крупный

из них, его диаметр составляет около 3274 миль (5268 километров), что делает его немногим больше Меркурия. Однако эти спутники вращаются вокруг Юпитера — планеты, которая намного больше Меркурия.

Однако, как говорит нам Wolfram|Alpha, подобный вывод неверен:



Всякий раз, когда дело доходит до конкретных данных (например, количественных), пусть даже в достаточно сырой форме, нам приходится иметь дело с «историей Wolfram|Alpha». Вот пример, вдохновленный давно любимым тестовым запросом Wolfram|Alpha о том, сколько индеек в Турции:

Расскажи мне о поголовье скота в Турции

Турция является крупным производителем домашнего скота, включая птицу, крупный рогатый скот, овец и коз. По данным Продовольственной и сельскохозяйственной организации Объединенных Наций (ФАО), по состоянию на 2020 год в Турции насчитывались следующие виды скота:

- Птица: в Турции хорошо развито птицеводство, поголовье составляет около 537 миллионов кур, 37 миллионов индеек и 3 миллиона уток.
- Крупный рогатый скот: в Турции относительно небольшое поголовье крупного рогатого скота, насчитывающее около 7,4 миллиона голов.
- Овцы: в Турции поголовье овец большое — около 25,6 миллиона овец.
- Козы: в Турции также имеется значительная популяция коз, насчитывающая около 10,7 миллиона коз.

Помимо этого, в Турции также есть меньшие популяции других животных, таких как лошади, ослы и верблюды.

На первый взгляд ответ опять кажется вполне правдоподобным; ChatGPT даже цитирует релевантный источник. Однако, как оказалось, эти данные были выдуманы:

Популяция скота в Турции
Крупный рогатый скот — 18 миллионов (на 2020 год)
Куры — 379 миллионов (на 2020 год)
Козы — 12 миллионов (на 2020 год)
Свиньи — 990 (на 2020 год)
Овцы — 42,1 миллиона (на 2020 год)

Тем не менее, что очень приятно, ChatGPT легко можно заставить «проверить факты»:

Дай мне ключевые слова для проверки количественного значения результата запроса
Количество куриц в Турции
Количество индеек в Турции
Количество уток в Турции
Количество крупного рогатого скота в Турции
Количество овец в Турции
Количество коз в Турции

Теперь запросите эти данные через API у Wolfram|Alpha:

```
In[ ]:= WolframAlpha[#, "SpokenResult"] &/@
{"Number of chickens in Turkey", "Number of turkeys in
Turkey",
"Number of ducks in Turkey", "Number of cattle in Turkey",
"Number of sheep in Turkey", "Number of goats in
Turkey"}
```

Out[ ]= {In 2020, Turkey had about 379 million chickens as livestock,
In 2020, Turkey had about 4.8 million turkeys as livestock,
In 2020, Turkey had 560000 ducks as livestock,
In 2020, Turkey had about 18 million cattle as livestock,
In 2020, Turkey had about 42.1 million sheep as livestock,
In 2020, Turkey had about 12 million goats as livestock}

Теперь мы можем попросить ChatGPT исправить первоначальный ответ, внедрив полученные данные (и даже обозначить исправленные данные жирным шрифтом):

*Вставь эти исправленные данные в исходный ответ, пометив каждое изменение знаком **:*
{«В 2020 году в Турции было около 379 миллионов кур», «В 2020 году в Турции было около 4,8 миллиона индеек», «В 2020 году в Турции было 560 тысяч уток», «В 2020 году в Турции было около 18 миллионов крупного рогатого скота», «В 2020 году в Турции было около 42,1 миллиона овец», «В 2020 году в Турции было около 12 миллионов коз»}

Здесь исправленные данные, где измененные части помечены как **:

Турция является крупным производителем домашнего скота, включая птицу, крупный рогатый

скот, овец и коз. По данным ФАО, по состоянию на 2020 год в Турции было зарегистрировано следующее поголовье скота:

- Птица: в 2020 году в Турции было около 379 миллионов кур, 4,8 миллиона индеек, а также около 560 тысяч уток.
- Крупный рогатый скот: в 2020 году в Турции было около 18 миллионов голов крупного рогатого скота.
- Овцы: в 2020 году в Турции было около 42,1 миллиона овец.
- Козы: в 2020 году в Турции было около 12 миллионов коз.

Помимо этого, в Турции также есть популяции других животных вроде лошадей, ослов и верблюдов.

Возможность вводить факты очень полезна, когда речь идет о вещах, связанных с данными или вычислениями в реальном времени (или в зависимости от местоположения и так далее). ChatGPT не сразу сможет ответить на вопросы, подобные следующим:

Какие планеты я могу увидеть сегодня вечером?

Извините, но у меня нет актуальной информации о том, какие планеты видны сегодня вечером

на небе. Чтобы узнать, какие планеты видны сегодня ночью, вам нужно будет воспользоваться специализированным приложением или веб-сайтом. Подобные сервисы используют ваше местоположение и текущее время, чтобы предоставить информацию о том, какие планеты видны на небе. Вот некоторые популярные приложения и веб-сайты: Sky Guide, Star Walk и Stellarium. Кроме того, вы можете проверить распечатанную карту неба или попросить астронома предоставить вам подробную информацию.

Однако вот что сможет предложить Wolfram|Alpha, если сделать запрос через интерфейс API:

Планеты | над горизонтом | 22:00,
Меркурий | Меркурий находится за горизонтом
Венера | Венера находится за горизонтом
Марс | Марс находится над горизонтом
Юпитер | Юпитер находится над горизонтом
Сатурн | Сатурн находится за горизонтом
Уран | Уран находится над горизонтом
Нептун | Нептун находится над горизонтом

Если мы введем эти данные в ChatGPT, он сгенерирует такой ответ:

Исправь этот ответ с учетом следующих данных:
Планеты | над горизонтом | 22:00,
Меркурий | Меркурий находится за горизонтом
Венера | Венера находится за горизонтом
Марс | Марс находится над горизонтом
Юпитер | Юпитер находится над горизонтом
Сатурн | Сатурн находится за горизонтом
С учетом новой информации можно сказать, что в 22:00 над горизонтом находятся планеты Марс, Юпитер, Уран и Нептун. Меркурий, Сатурн и Венера находятся ниже горизонта. Видимость планет может меняться в зависимости от вашего местоположения и времени года. Я рекомендую использовать специализированные приложения или веб-сайт, чтобы получить наиболее точную и актуальную информацию о том, какие планеты видны из вашего местоположения.

Иногда происходит интересное взаимодействие между вычислительными инструментами и «человеческими» способностями. Вот довольно причудливый вопрос, заданный Wolfram|Alpha (и последний даже проверяет, хотите ли вы мягкое мороженое):

Вопрос: сколько калорий находится в кубическом световом годе мороженого?
Интерпретация ввода
Мороженое количество 1 ly3 (кубический световой год) общее количество калорий

Средний результат
1,4 × 1054 кал (диетические калории)

Единицы измерения
5,8 x 1057 Дж (джоули)
5,8 × 1064 эрг (единица официально устарела)

ChatGPT сначала немного запутался в понятии объема:

Сколько калорий в кубическом световом году мороженого?
Прошу прощения, но невозможно измерить кубический световой год чего-либо, не говоря уже о мороженом. Световой год — это единица расстояния, а не объема, и она равна примерно 5,88 триллиона миль. Это мера расстояния, которое свет может преодолеть за один год. Следовательно, невозможно измерить кубический световой год вещества, включая мороженое.

Но затем он, кажется, понимает, что столько мороженого — это довольно глупо:

Кубический световой год — чрезвычайно большая единица объема. Фактически она настолько большая, что в объем такого размера невозможно уместить ни одно известное вещество, включая мороженое.

Для справки: световой год — это единица измерения расстояния, равная расстоянию, которое свет проходит за один год. Расстояние, которое свет проходит за год, составляет примерно 5,88 триллиона миль (9,46 триллиона километров).

Учитывая размеры светового года, становится ясно, что кубический световой год — это невероятно большая единица объема. Поэтому подсчитать количество калорий в кубическом световом году мороженого не представляется возможным, так как уместить столько мороженого в один объем просто невозможно.

ПЕРСПЕКТИВЫ

За прошедшее десятилетие машинное обучение добилось невероятных успехов, к числу которых относится и ChatGPT. Распознавание изображений, перевод устной речи в текст, перевод с одного языка на другой — в каждой из этих областей успех был достигнут совершенно внезапно, так что задачи превратились из практически невыполнимых во вполне выполнимые.

Однако результаты никогда не бывают безупречными. Что-то отлично работает только в 95% случаев, и, как бы мы ни старались, все равно остаются 5% неудач. Иногда это означает полный провал. Однако существуют задачи, для которых 95% успеха — это достаточно хорошо. Возможно, причина в том, что не существует «правильного» ответа. Или в том, что кто-то пытается выявить возможности, часть из которых человек — или алгоритм — затем использует и адаптирует.

Примечательно, что нейронная сеть с несколькими сотнями миллиардов параметров, генерирующая текст по токенам, способна делать то же самое, что и ChatGPT. Учитывая этот невероятный — и неожиданный — успех, можно подумать, что если просто обучить сеть достаточному количеству вещей, то она сможет делать что угодно. Однако это так не работает. Фундаментальные знания о машинных вычислениях — и особенно концепция вычислительной несводимости — ясно дают понять, что это невозможно. Самое важное мы уже видели, наблюдая за развитием машинного обучения. Прорыв, подобный ChatGPT, случится еще раз. И он будет еще мощнее. В будущем такие механизмы не будут останавливаться перед тем, что раньше считалось невозможным.

Да, ChatGPT продолжит помогать людям писать тексты, вносить предложения и генерировать сырой текст для различных целей. Однако, когда дело касается вещей, которые задумываются как идеальные, оказывается, что машинное обучение (как, собственно, и люди) здесь не подходит.

Пример этого мы видели выше. ChatGPT отлично справляется с человеческими запросами, которые не требуют «правильного ответа». Но когда нужен четкий и конкретный ответ, чат-бот теряется. И все же

существует отличный способ решить эту проблему: подключить ChatGPT к Wolfram|Alpha и всем его «суперспособностям» в области вычисляемых знаний.

Внутри Wolfram|Alpha все превращается в вычислительный язык и в точный код Wolfram Language, который на каком-то уровне является идеальным. Тем не менее важно заметить, что ChatGPT не обязан его генерировать. Он продолжит генерировать естественный язык, а затем Wolfram|Alpha сможет перевести этот естественный язык в точный язык Wolfram Language.

В каком-то смысле можно утверждать, что ChatGPT никогда «по-настоящему не понимает» вещей; он просто «знает», как производить полезное. В случае с Wolfram|Alpha ситуация немного другая. Дело в том, что, как только Wolfram|Alpha преобразует что-то в Wolfram Language, он получает полное, точное и формальное представление, на основе которого можно выполнять надежные вычисления. Излишне говорить, что существует множество вещей, для которых у нас нет формальных вычислительных представлений, хотя мы все равно можем говорить о них (пусть, возможно, и неточно) на естественном языке. И для этого у нас есть ChatGPT, обладающий весьма впечатляющими возможностями.

Но, подобно людям, ChatGPT иногда нуждается в более формальной и точной помощи. Однако ему не обязательно быть «формальным и точным», сообщая нам, чего он хочет, потому что Wolfram|Alpha может общаться с ним на родном языке ChatGPT — естественном языке. А Wolfram|Alpha позаботится о «формальности и точности» при переходе на свой родной язык — Wolfram Language. Подобная структура обладает большим потенциалом.

Этот потенциал заметен не только на уровне типичных чат-ботов или приложений для генерации текста. Он распространяется и на такие вещи, как наука о данных, моделирование и симуляция, анализ данных и так далее. В каком-то смысле это прямой способ получить лучшее из обоих миров — «человеческого» мира ChatGPT и мира точных вычислений Wolfram Language.

А что, если ChatGPT выучит Wolfram Language? В каком-то смысле это возможно, и, более того, это уже началось. В конце концов, я думаю, в будущем ChatGPT будет работать непосредственно в Wolfram Language. Это интересная и уникальная ситуация, когда благодаря полномасштабному вычислительному языку Wolfram Language ИИ сможет говорить о многих вещах вычислительными терминами.

Вся концепция Wolfram Language заключается в том, чтобы представлять обыденные вещи в виде вычислений. Обычные языки программирования помогают ставить перед компьютерами точные задачи. Wolfram Language — в качестве полномасштабного вычислительного языка — представляет собой нечто гораздо большее. По сути, он задуман как язык, на котором и люди, и компьютеры смогут мыслить путем вычислений.

Много столетий назад, когда математические обозначения только появились, они упростили математическое мышление. Их изобретение вскоре привело к появлению алгебры, математического исчисления и в итоге других математических наук. Цель Wolfram Language — сделать что-то подобное для вычислительного мышления, но теперь не только для людей, и открыть все неизвестные дисциплины, которые могут быть открыты вычислительной парадигмой.

Мне самому Wolfram Language принес много пользы именно в качестве «языка для мышления». Было невероятно занимательно наблюдать, как за последние несколько десятилетий мы добились такого количества результатов именно благодаря тому, что люди мыслили путем вычислений с помощью Wolfram Language. Так что же насчет ChatGPT? То же самое может произойти

и с ChatGPT, однако я пока не знаю, как именно это будет работать. Но речь не о том, чтобы ChatGPT научился выполнять вычисления, которые уже умеет делать Wolfram Language. Речь о том, чтобы ChatGPT научился использовать Wolfram Language так, как это делают люди. Чтобы ChatGPT придумал аналог «творческих эссе», но написанных уже не на естественном, а на вычислительном языке.

Я давно обдумываю концепцию компьютерных текстов, написанных людьми, но сочетающих в себе как естественный, так и компьютерный язык. Вопрос заключается в том, сможет ли ChatGPT написать эти тексты — и сможет ли он использовать Wolfram Language, чтобы обеспечить содержательное общение не только с людьми, но и с компьютерами. Существует потенциально интересный цикл обратной связи, включающий в себя фактическое выполнение кода Wolfram Language. Но решающим моментом является то, что богатство и технологичность «идей», представленных кодом Wolfram Language, в отличие от обычного языка программирования, ближе к тому, с чем ChatGPT волшебным образом сумел разобраться на естественном языке.

Другими словами, Wolfram Language, как и естественный язык, является достаточно выразительным, чтобы

на нем можно было задавать промпты для ChatGPT. Да, Wolfram Language можно запускать непосредственно на компьютере. Но в качестве промпта ChatGPT есть возможность его использовать для «выражения идеи», историю которой нужно продолжить. Он способен описывать некую вычислительную структуру, оставляя на долю ChatGPT обдумывать то, что можно сказать в вычислительном отношении об этой структуре, которая — согласно информации, полученной благодаря людям, — была бы им интересна.

Неожиданный успех ChatGPT открыл перед нами множество захватывающих перспектив. И сейчас у нас есть возможность наделить ChatGPT сверхспособностями в области вычисляемых знаний с помощью Wolfram|Alpha. Таким образом, он может не просто выдавать реалистичный результат, подобный тому, что мог бы сделать человек, а использовать вычисления и знания, которые есть у Wolfram|Alpha и Wolfram Language.

ДОПОЛНИТЕЛЬНЫЕ РЕСУРСЫ

Стивен Вольфрам, «Машинное обучение для учащихся средних школ».
Краткое введение в основные понятия машинного обучения:
wolfr.am/ML-for-middle-schoolers.

Этьен Бернард, «Введение в машинное обучение».
Руководство по современному машинному обучению с напечатанным работоспособным кодом:
wolfr.am/IML-book; онлайн-версия: wolfr.am/IML.

Wolfram Machine Learning.
Возможности машинного обучения в Wolfram Language:
wolfr.am/core-ML.

Machine Learning at Wolfram U.
Интерактивные занятия и курсы по машинному обучению на разных уровнях:
wolfr.am/ML-courses.

Стивен Вольфрам, How Should We Talk to AIs?
Краткое эссе 2015 года об общении с ИИ на естественном и вычислительном языке:
wolfr.am/talk-AI.

Wolfram Language:
wolfram.com/language.

Wolfram|Alpha:
wolframalpha.com.

Онлайн-ссылка на все ресурсы:
wolfr.am/ChatGPT-resources.

Научно-популярное издание
Цифровые технологии

Вольфрам Стивен

Как устроен ChatGPT?

Полное погружение в принципы работы и спектр возможностей самой известной нейросети в мире

Руководитель редакционной группы *Светлана Мотылькова*
Шеф-редактор *Ксения Свешникова*
Ответственный редактор *Ксения Борисова*
Литературный редактор *Дарья Балтрушайтис*
Креативный директор *Яна Паламарчук*
Арт-директор *Антон Героев*
Дизайн обложки *Антон Героев*
Верстка *Владимир Снеговский*
Корректоры *Евлалия Мазаник, Людмила Широкова*

Подписано в печать 24.05.2024
Формат 60×88/16. Гарнитура Caslon 540.
Усл. печ. л. 11,73. Тираж 3000 экз.
Заказ 4655.

ООО «Манн, Иванов и Фербер»
123104, Россия, г. Москва, Б. Козихинский пер., д. 7, стр. 2

mann-ivanov-ferber.ru
vk.com/mifbooks

Отпечатано в АО «Первая Образцовая типография»,
филиал «УЛЬЯНОВСКИЙ ДОМ ПЕЧАТИ»
432980, Россия, г. Ульяновск, ул. Гончарова, д. 14
uldp.ru

16+

ChatGPT стремительно завоевал популярность благодаря способности «понимать» речь, действовать в ответ на промпт, словно разумный, и найти ответ на любой вопрос. Но как именно нейросеть это делает?

Книга выдающегося ученого Стивена Вольфрама расскажет, что находится под капотом большой языковой модели ChatGPT и почему она способна выполнять действия, которые раньше были под силу лишь человеку.

Вы узнаете:
— сколько нейронов нужно для выполнения конкретной задачи;
— каким образом концепция эмбеддингов реализуется в ChatGPT;
— как обучаются нейронные сети;
— многое другое о внутренних процессах ChatGPT.

Стивен Вольфрам — удостоенный наград математик, физик и программист, а также создатель программных систем, таких как Wolfram|Alpha. Более 35 лет он был генеральным директором глобальной технологической компании Wolfram Research и, кроме того, отвечал за ряд новаторских достижений в области фундаментальной науки.

* * *

«Это лучшее объяснение того, что делает ChatGPT, которое я видел».

СЭМ АЛЬТМАН,
генеральный директор OpenAI

9 785002 146048

МИФ mann-ivanov-ferber.ru @mifbooks