Inteligência “Artificial”

MC102

Termos

Antes de começarmos

IA não possui significado algum

- qualquer coisa que o humano médio acha que uma calculadora não consegue fazer é IA
- televisão, fogão, geladeira...

Machine Learning matemática glorificada e estatística com um nome que soa melhor.

- em vez de você programar as regras exatas na mão (com uma infinidade de `if` e `else`), você joga uma montanha de dados num modelo de otimização e deixa ele descobrir o padrão e ajustar os parâmetros sozinho.
- a máquina não "aprende", ela minimiza erros.

LLM (Large Language Model) o "autocompletar" do teclado do seu celular, só que com anabolizantes e depois de ter lido a internet inteira.

- não tem consciência, não raciocina e não compreende o que você fala.
- ele apenas usa probabilidade e álgebra linear para chutar qual é a próxima palavra que faz mais sentido gerar com base no contexto anterior.

Token é a unidade básica de texto que o modelo de fato “mastiga”.

- computadores não entendem palavras, eles entendem números. Um token é um pedaço de texto (pode ser uma palavra inteira, uma sílaba ou apenas uma letra) mapeado para um número.
- para a máquina, “sol” pode ser um token só, mas “inconstitucionalmente” vai ser quebrado em vários bloquinhos. É a “moeda” que dita o limite de memória e o custo de rodar o modelo.

História

1948 :: Information as a thing you can measure

- Quanta informação uma sentença carrega?
- Toda informação é um fluxo de 1s e 0s
- Entropia
- Quão surpreso você fica com a próxima letra/palavra?
  - Quando é fácil de adivinhar -> baixa informação
  - Quando é difícil/surpreendente -> alta informação
- Chutar o próximo **token**/palavra é o que a LLM faz

1957 :: The Perceptron

- Inspirado pela forma como os neurônios funcionam
- Recebe uma **input**, atribui pesos a ela e, em caso de erro, altera esses pesos para “aprender”

1969 :: PerceptronS

- Uma única camada de **perceptrons** não consegue aprender conceitos lógicos simples como XOR
- Mas camadas de **perceptrons** conseguem!
- Porém, ninguém sabia como treinar isso
  - Com um único **perceptron**, você pode simplesmente alterar os pesos daquele neurônio
  - Mas, quando você tem múltiplas camadas, como saber se o último neurônio é o problema?
- Neural Network

1986 :: Back-propagation

- Executar os dados para frente (**forward**) através da **neural network**
- Medir o quão errado o resultado está (**loss-function**)
- Empurrar o erro para trás (**backwards**) através de cada camada, usando a **chain rule** do cálculo para ajustar cada camada em uma direção que a torne um pouco menos errada
- A grande surpresa é que as camadas intermediárias (**hidden layers**) começam a formar conceitos nos quais não foram treinadas explicitamente
  - Por exemplo, suponha que treinamos uma **neural network** para identificar pizza de pepperoni
  - As **hidden layers** podem começar a "aprender" como um círculo é representado
  - E ela pode ser capaz, mais tarde, de identificar rodas em uma imagem
  - Já que ela possui o "conceito" de círculo em uma imagem 2D
- Essa ideia é central para o que fazemos hoje, mas como cada ajuste é apenas um pequeno passo na direção correta em múltiplas camadas, precisamos de MUITOS dados, o que não existia na época.

1998 :: E assim fez-se a Internet

- O que resolve o problema dos “dados”.
- Para um humano ler tudo que o ChatGPT-3 (o original, de 2020) usou para ser treinado, seriam necessários **2300 anos** sem parar, 24h todos os dias.

2012 :: ImageNet

- **Backpropagation** precisa de dados + poder computacional
- Na verdade, também precisa da arquitetura “certa”

2017 :: Attention is all you need

- Naquela época, já era possível ter **Language Models** que previam a próxima palavra, mas eles frequentemente "esqueciam" o que estavam dizendo e se perdiam.
  - TODO: dar um exemplo
  - Já que liam e previam **tokens** sequencialmente
- Adicionar uma camada que permite que cada palavra influencie o resultado de todas as outras palavras, de uma forma não sequencial.
- Isso é chamado de **transformer**
  - Principalmente, isso permite paralelizar o processo
  - Para isso, utilizaremos processadores extremamente eficientes em fazerem multiplas coisas ao mesmo tempo: GPU
- Cada palavra (token) é mapeada à um vetor (processo chamado de **embedding**):
  - Esse mapeamento é também "aprendido" de forma similar
- Quando estamos processando as frases "a manga da camisa" e "comprei uma manga na frutaria", originalmente, a palavra "manga" teria o mesmo vetor

- O que a operação de “atenção” faz é permitir que “camisa” afete o vetor de “manga”, modificando sua direção para algo levemente diferente do que “manga” seguido de “frutaria”.
- E esse processo se repete

2020 :: LM are few-shot learners

- “Intelligence isn’t some secret algorithm, but rather it simply emerges once you cross a threshold of scale.”

Dicas gerais para utilizar LLMs

Não há “raciocínio” ou “pensamento”

- Não há NENHUM compromisso com a lógica ou a razão.
- A cada previsão, cada token adicionado à sua resposta, a LLM está refazendo a conta para cada um dos tokens em todo o histórico do chat (contexto).
  - Isso requer um poder computacional absurdo
- Ao invés de prever uma única palavra, como o que a gente vê, o que ele faz é predizer uma função de probabilidade, uma distribuição de chance, de qual será a próxima.
  - Baseado nisso, ele escolhe qual que ele vai realmente inserir e repete o processo.
- Assim, apesar do modelo em si ser determinístico, o resultado é “aleatório”.
  - Token improváveis, ao serem selecionados, dão um ar mais natural à resposta.

Não há compromisso ou responsabilidade

- Apesar de tudo que discutirmos ter sido “desenhado” por humanos, o exato valor de cada parâmetro não é.
- Dado o fato de serem bilhões desses “pesos”, a tarefa de entender o porquê de uma resposta em específico é impossível.
  - Mas e se a LLM fizer merda? Quem é o responsável?
  - Quantas das decisões queremos que ela realmente tome?

Dicas

- Só utilizem com coisas que são “fáceis” de confirmar se é verdade.
  - Código é um bom candidato: se eu pergunto “como fazer a média dos valores de uma coluna nessa tabela”, ele vai me responder com *algo* que eu posso só copiar e ver se deu certo ou não.
  - Cuidado com coisas que não é viável ou prático confirmarmos: “qual o melhor remédio para dor de cabeça?”
- Tarefas manuais repetitivas
  - Estas, geralmente, poderiam ser resolvidas com um código bem feito
  - Mas é mais prático colocar para a LLM fazer…
  - Conferir se ela fez certo é “fácil” se o texto não for muito grande
- Pesquisar a existência de conceitos que você consegue depois ir a atrás
  - Por exemplo, digamos que eu preciso fazer um aplicativo web que vai receber alguns campos de texto e devolver um PDF.
  - Eu não sei que métodos, tecnologias eu preciso para isso
  - Posso perguntar para a LLM e ela, muitas vezes, voltará com termos e associações que eu não conhecia.

- Não deixe que ela tome a decisão, mas que ela proponha a relação com conceitos.

Principal erro profissional

1. Débito tecnológico
   - Tu tá numa empresa
   - Tem uma tarefa complexa, chata e repetitiva -> “wow, código!”
   - Tu começa a pedir para uma LLM produzir uma solução em código, que tu não faz ideia do que está acontecendo
     - Seja porque você não entende o código
     - Seja porque é grande de mais e é inviável tomar conta de tudo
   - Não é uma questão de se vai dar B.O., mas quando vai dar.
   - Quando der, sua única chance é voltar na LLM
   - Eventualmente ela irá errar, eventualmente ela não vai conseguir resolver.
2. *Over-engeneering*
   - Lembrem que LLMs são a média da internet
   - O que existe na internet?
     - Ou é babozeira e ideia ruim
     - Ou é a solução para problemas ultra complicados

- Chances são que a LLM vai super complicar o seu problema e te dar uma resposta mal codada
  - Ou você realmente acha que a média dos código soltos pelo mundo é boa???

3. Culpabilidade
   - Troque LLM por “o estagiário”.
   - O quanto da sua empresa, da sua profissão você está disposta a deixar na mão de um estagiário?
   - Por mais brilhante que ele seja, você está pronta para arcar com as consequências de quando ele falhar?
   - E se é ele que está “trabalhando”, se é ele que está se “dedicando”, você é realmente necessária?

Por fim, ética

1. Model colapse
2. A moral de grandes *datacenters*
3. Quem deixou utilizarem meus dados?
4. É realmente inevitável?

Fontes e Recomendações

Fontes

- I read every major CS paper of the last 100 years... - YouTube - Um breve (10min) resumo dos últimos 100 anos de computação, focado apenas no background da atual IA

Canal Veritassium:

- Origem do conceito teórico de computadores: https://youtu.be/HeQX2HjkcNo
- Como fabricamos processadores: https://youtu.be/MiUHjLxm3V0
- Origem dos computadores: https://youtu.be/FU_YFpfDqqA
- Cadeias de Markov é um conceito importante da Probabilidade e Estatística para boa parte das "predições" feitas pelos computadores: https://youtu.be/KZeIEiBrT_w

3blue1brown:

- Selected lectures on computational thinking for MIT 18.S191 - YouTube
- Grande parte da tecnologia (e intuição) por trás do aprendizado de máquina moderno e LLMs: Neural networks - YouTube
- Computação Quantica: But what is quantum computing? (Grover's Algorithm) - YouTube
- Bitcoin! $$$: But how does bitcoin actually work? - YouTube

- Detecção (e correção) de erro: esse é um pequeno problema (de grande impacto) que é resolvido de uma maneira "esperta" até hoje: https://youtu.be/X8jsijhllIA https://youtu.be/b3NxrZOu_CE
- Algebra Linear é a base de toda a matemática da computação: https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab

Outros:

- Componentes eletrônicos de um computador (portas lógicas): https://www.youtube.com/watch?v=QZwneRb-zqA
- Como os transistores funcionam? https://www.youtube.com/watch?v=_Pqfjer8-O4