

Under the Hood of AI Systems

Ramon Béjar

16 June 2023

GREiA - UdL



Artificial Intelligence and Alan Turing

In 1950, in the work: "Computing Machinery and Intelligence" Alan Turing presented an interesting approach to the question:

"Can Machines Think?"

Given the difficulties to find acceptable definitions of *machine* and *think*, he proposed to change the question to a new one:

Can a digital computer win **the Imitation Game?**

Artificial Intelligence and Alan Turing

The Imitation Game

We have three players:

A: A machine

B: A woman

C: An interrogator (either sex)

The interrogator stays in a room apart from the other two. The object of the game for the interrogator is to determine which of the other two is the machine and which is the woman. Both A and B try to convince C that they are the woman, through a series of questions (by C) and answers, always typewritten.

Are there imaginable digital computers which would do well in the imitation game?

Artificial Intelligence and Alan Turing

The belief of Alan Turing:

I believe that in about fifty years' time it will be possible to programme computers, with a storage capacity of about 10^9 , to make them play the imitation game so well that an average interrogator will not have more than 70 per cent, chance of making the right identification after five minutes of questioning.

The original question, 'Can machines think!' I believe to be too meaningless to deserve discussion

The foundations of AI Research: Dartmouth 1956

In 1956, a small group of scientists gathered for the Dartmouth Summer Research Project on Artificial Intelligence, which was the birth of this field of research.

Goal of the conference:

"to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it"

So, "describe" to be able to "simulate"

In 1965, Joseph Weizenbaum implemented an AI system, called Eliza, that caused quite a sensation in his time. Eliza was explained in this paper:

"ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine"

From the Introduction of the Eliza paper:

"But once a particular program is unmasked, once its inner workings are explained in language sufficiently plain to induce understanding, its magic crumbles away; it stands revealed as a mere collection of procedures, each quite comprehensible."

Can we perform this **unmasking** for every AI system?

Talking with Eliza

Eliza allowed **certain kinds** of *natural language* conversation between a man and a computer.

Conversation example

C: How do you do. please tell me your problem

P: I am desperate

C: Is it because you are desperate that you came to me

P: yes

C: you seem quite positive

P: Men are all alike

C: In what way

P: Are you listening me?

C: Why are you interested in whether I am listening you or not

...

Under the hood of Eliza

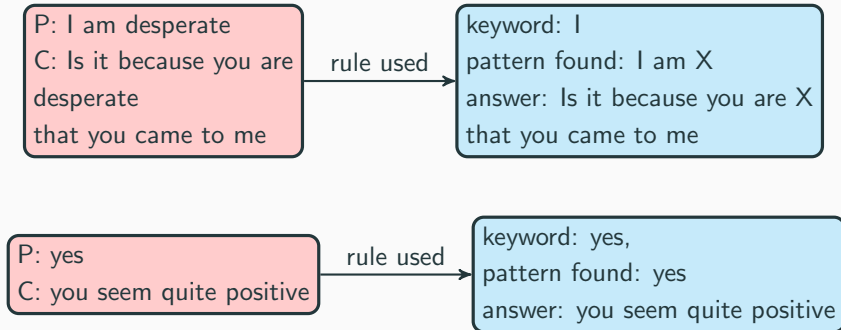
To allow different kinds of conversation, ELIZA uses a set of rules that follow the following general idea:

If query MATCHES a certain pattern then
TRANSFORM the query to generate an answer

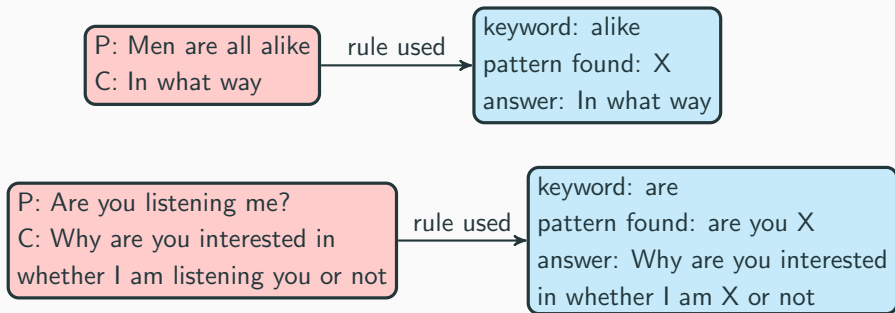
The rule to use is determined by the existence of certain keywords in the query of the person.

If different keywords are present in the same query, a preference order is used.

Under the hood of Eliza - Conversation Example



Under the hood of Eliza - Conversation Example



note: "me" is always transformed to "you"

Eliza and the Turing Test

Eliza was probably the first attempt to build an AI system that was able "to chat" with a person in a very "human-like" style.

What is Eliza really doing?

- Is Eliza thinking ?
- Or is she cheating ? (it makes you believe it understands what you are saying?)

At least, answers were syntactically correct.

But one would say it would not pass a "Turing Test".

Symbolic Mathematics - Symbolic Integration

Consider these rules for solving integrals:

$$\int dx = x$$

$$\int x^n dx = \frac{x^{(n+1)}}{(n+1)}$$

$$\int e^x dx = e^x$$

You can use them to solve some integrals **without** really understanding what an integral really is !

Symbolic Mathematics - Symbolic Integration

The previous rules are direct rules: a direct solution for each problem

But some rules for solving integrals do not give you a direct answer, they give you a way to solve the problem by solving some other more simple problems, like **integration by parts**:

$$\int (f(x) \cdot g(x))dx = f(x) \int g(x)dx - \int \left(\int g(x)dx \right) \cdot f'(x)dx$$

That is, to solve the integral $\int (f(x) \cdot g(x))dx$, you need to :

1. P1: Solve $f'(x)$

Symbolic Mathematics - Symbolic Integration

The previous rules are direct rules: a direct solution for each problem

But some rules for solving integrals do not give you a direct answer, they give you a way to solve the problem by solving some other more simple problems, like **integration by parts**:

$$\int (f(x) \cdot g(x))dx = f(x) \int g(x)dx - \int \left(\int g(x)dx \right) \cdot f'(x)dx$$

That is, to solve the integral $\int (f(x) \cdot g(x))dx$, you need to :

1. P1: Solve $f'(x)$
2. P2: Solve the integral $\int g(x)dx$

Symbolic Mathematics - Symbolic Integration

The previous rules are direct rules: a direct solution for each problem

But some rules for solving integrals do not give you a direct answer, they give you a way to solve the problem by solving some other more simple problems, like **integration by parts**:

$$\int (f(x) \cdot g(x))dx = f(x) \int g(x)dx - \int \left(\int g(x)dx \right) \cdot f'(x)dx$$

That is, to solve the integral $\int (f(x) \cdot g(x))dx$, you need to :

1. P1: Solve $f'(x)$
2. P2: Solve the integral $\int g(x)dx$
3. P3: Finally, solve $\int \left(\int g(x)dx \right) \cdot f'(x)dx$

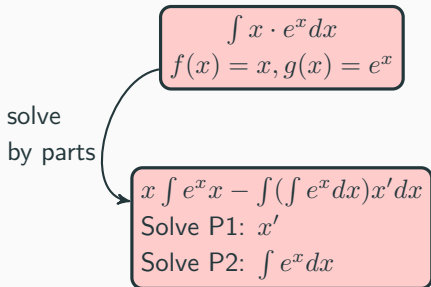
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$

$$\int x \cdot e^x dx$$
$$f(x) = x, g(x) = e^x$$

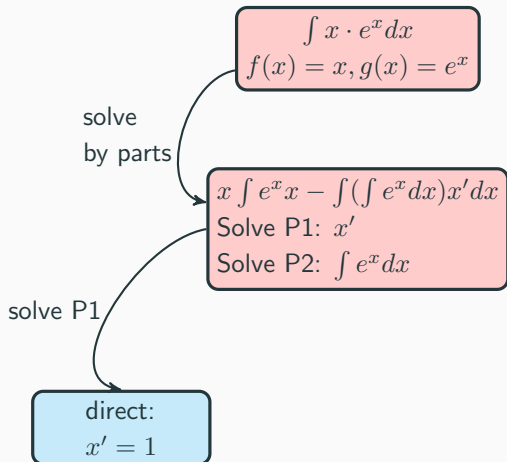
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



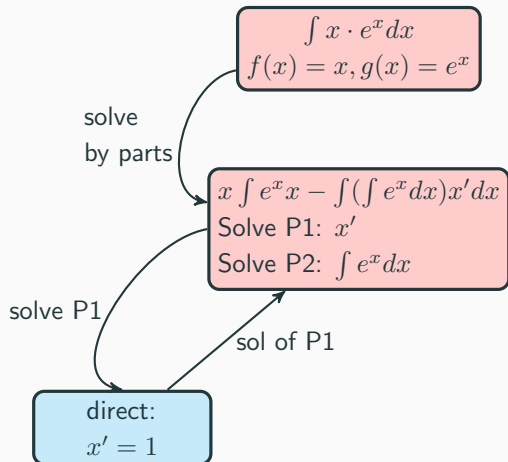
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



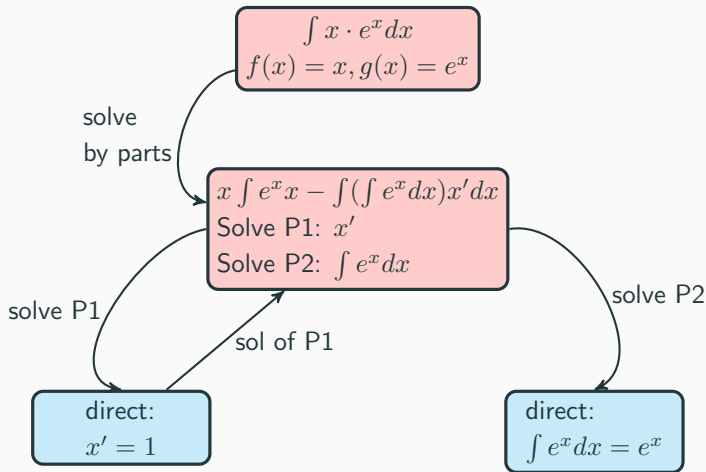
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



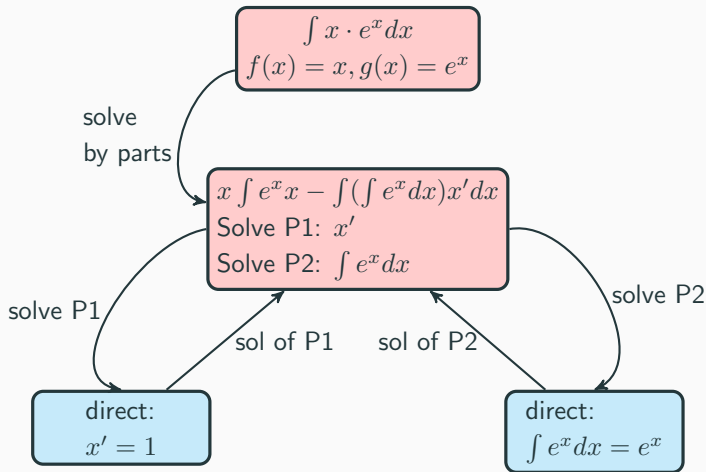
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



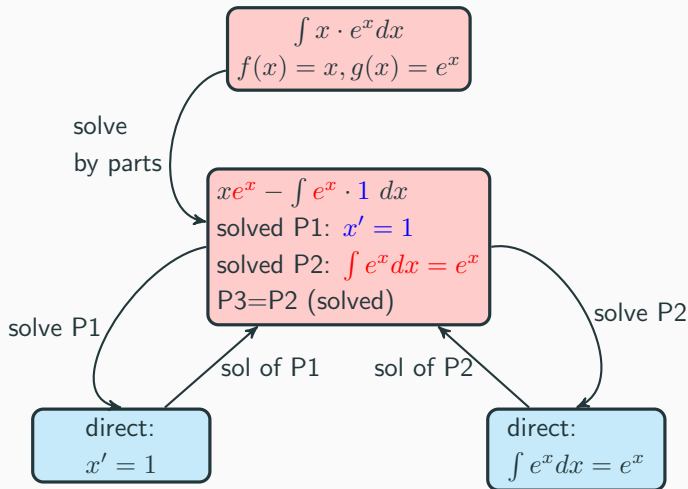
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



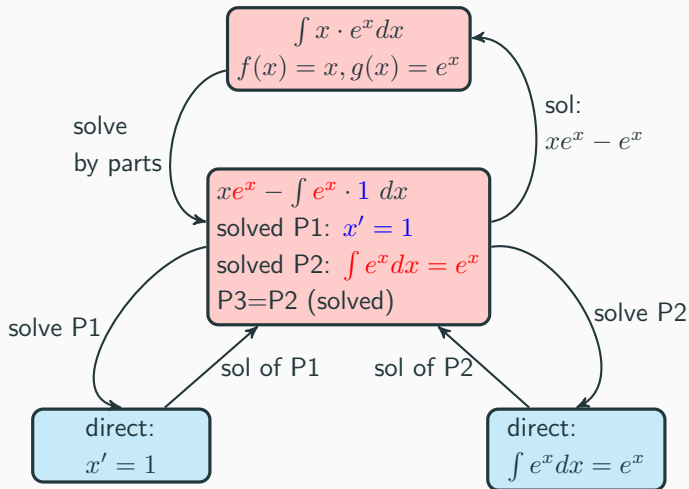
Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



Symbolic Mathematics - Symbolic Integration

Consider for example the integration by parts of $\int x \cdot e^x dx$



Symbolic Mathematics - Symbolic Integration in SymPy

Solving $\int x \cdot e^x dx$ in SymPy

Not only solves it, but it can **explain** the solving steps !

Integral Steps:

```
integrate(x*exp(x), x)
```

Fullscreen

1. Use Integration by parts:

$$\int u \, dv = u \, v - \int v \, du$$

Let $u(x) = x$ and let $dv(x) = e^x$.

Then $du(x) = 1$.

To find $v(x)$:

- A. The integral of the exponential function is itself.

$$\int e^x \, dx = e^x$$

Now evaluate the sub-integral.

2. The integral of the exponential function is itself.

$$\int e^x \, dx = e^x$$

3. Now simplify:

$$(x - 1) e^x$$

4. Add the constant of integration:

$$(x - 1) e^x + \text{constant}$$

Symbolic Mathematics - Symbolic Integration

These symbolic integration rules follow the following general idea:

direct integral ($\int e^x dx$):

If integral MATCHES a certain pattern then
TRANSFORM the query to generate a DIRECT answer

Symbolic Mathematics - Symbolic Integration

These symbolic integration rules follow the following general idea:

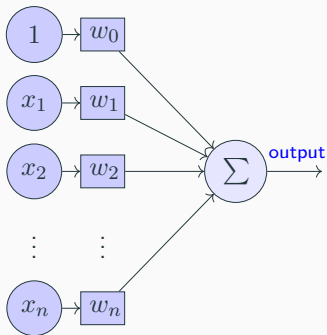
direct integral ($\int e^x dx$):

If integral MATCHES a certain pattern then
TRANSFORM the query to generate a DIRECT answer

NOT direct integral ($\int x \cdot e^x dx$):

If integral MATCHES a certain pattern then
TRANSFORM the PROBLEM INTO subproblems
SOLVE each subproblem
TRANSFORM solutions of subproblems to generate final answer

The simplest artificial neural network: the Perceptron



inputs weights

Consider an input vector $[x_0 = 1, x_1, x_2, \dots, x_n]$ representing a **question** to our system. In the original perceptron, the **output** value is a binary value (1/0) decided as:

$$\text{output} = \begin{cases} 1 & \text{if } \sum w_j x_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

But we can also use other final functions in the **output**, like for example using the same linear input combination ($\sum w_j x_j$) as the output value.

Or some nonlinear function:

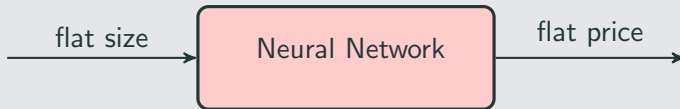
$$\text{output} = \begin{cases} \sum w_j x_j & \text{if } \sum w_j x_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

The weight value w_i represents how relevant is the input feature x_i to decide the final value of the perceptron.

Neural Network Models: Example 1

Predicting the price of a flat by its size

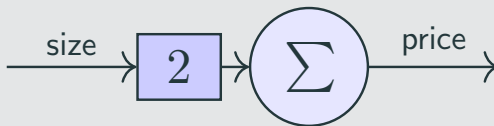
Consider a set of four flats with their size and price. Can we think about a simple neural network model to predict the price?



size	price
20	40
30	60
60	126
100	206

Neural Network Models: Example 1

Predicting the price of a flat by its size

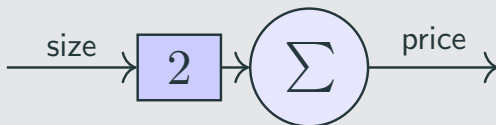


$$price(size) = 2 * size$$

But this model gives an approximate value for the table we have seen before, not the exact price!

Neural Network Models: Example 1

Predicting the price of a flat by its size - checking the model



size	real price	predicted value	error
20	40	40	0
30	60	60	0
60	126	120	6
100	206	200	6

Neural Network Models: Example 2

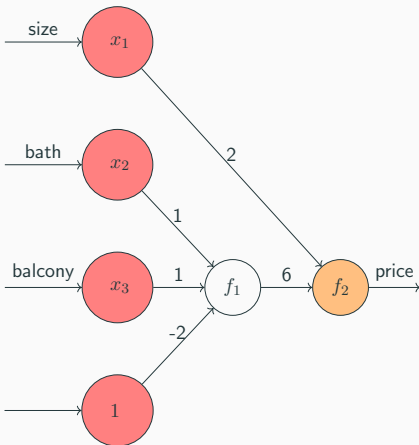
Consider now also whether the flat has two or more baths (boolean value) and one or more balconies (boolean value)

size	baths ≥ 2	balconies ≥ 1	price
20	1 (true)	0 (false)	40
30	0 (false)	1 (true)	60
60	1 (true)	1 (true)	126
100	1 (true)	1 (true)	206

With this new information, it may be possible to obtain a more complex model with better predictions

Neural Network Models: Example 2

What is the function computed by this neural network ?



1. First, the inner node computes the function f_1 :

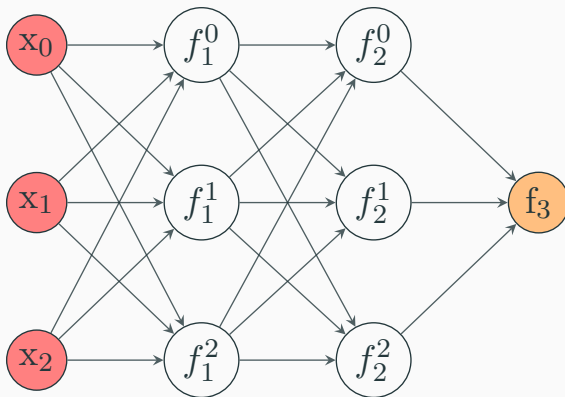
$$f_1(x_2, x_3) = \begin{cases} 1 & \text{if } x_2 + x_3 \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

2. Second, the final node computes the function f_2 :

$$f_2(x_1, f_1(x_2, x_3)) = 2x_1 + 6f_1(x_2, x_3)$$

that is, the price is equal to two times the size, plus 6000 euros when it has balcony and at least two bathrooms

Neural Network Models: More layers, More Complex Functions



We can say that :

- f_3 is composition of $f_2 = [f_2^0, f_2^1, f_2^2]$
- f_2^i is a composition of $f_1 = [f_1^0, f_1^1, f_1^2]$
- f_1^i is a composition of $x = [x_0, x_1, x_2]$

Neural Network Models: Learning a Model from Data

We can automatically learn a NN model given a data set of correct samples $(s_0, f(s_0)), \dots, (s_m, f(s_m))$ of the function we want to learn:

1. Initialize all the parameters (weights between neurons) at random
2. Compute the error for each sample of your data , then its average error
3. while you do not have a good enough model:
 - 3.1 Compute the partial derivative of the error with respect to each parameter w_i : $\frac{\partial Error}{\partial w_i}$
 - 3.2 Modify each parameter w_i proportionally to $-\frac{\partial Error}{\partial w_i}$
 - 3.3 Update the Error value

Talking with a Neural Network?

Remember, Eliza was probably the first attempt to build an AI system that was able "to talk" with a person in a very "human-like" style.

The idea behind the GPT family of models (GPT-1, GPT-2, GPT-3, GPT 3.5...) is to have neural networks that are able to:

1. Answer questions to answers within a broad domain of knowledge, in an acceptable way. This would be a **pretrained model**.
2. Then, starting from a pretrained model, we refine it through a more concrete learning process with the goal to answer questions for a more specific problem (e.g. translate from English to Catalan) in a very reliable way.

Generative Pretrained Transformers (GPT)

In a neural network based on the GPT architecture, the goal is to predict, given an input sequence of words:

$$input = [\text{Who discovered America?}]$$

the right **next word** (or the most correct one, if many could be considered correct) So:

$$input = [\text{Who discovered America?}] \rightarrow answer = [\text{Columbus}]$$

Generative Pretrained Transformers (GPT)

Being more accurate, what a GPT model computes is a conditional probability distribution:

$$Pr(\text{next word } j = w \mid \text{previous words} = [w_{j-k}, w_{j-(k-1)}, \dots, w_{j-1}])$$

where k is the size of the **context window**, and important parameter in GPT. In GPT-3: $k = 2048$

The conditional distribution is defined over a vocabulary of 50,257 tokens (subwords) on GPT-3

Generative Pretrained Transformers (GPT) - Long Answers

GPT is able to generate long answers (more than one word), using the following **autoregressive process**:

1. Input: [Sing the song Waterloo:] ,
GPT highest prob. answer: [My,]
2. Input: [Sing the song Waterloo: My,] ,
GPT highest prob. answer: [my]
3. Input: [Sing the song Waterloo: My, my] ,
GPT highest prob. answer: [At]
4. Input: [Sing the song Waterloo: My, my At] ,
GPT highest prob. answer: [Waterloo,]
5. ...

Generative Pretrained Transformers (GPT) - Long Answers

But because the output of GPT is really a probability distribution, we can ask to generate different answers by sampling from its probability distribution, instead of always picking the word with highest probability:

Input: [Who discovered America?]

1. GPT highest prob. answer: [Christopher Columbus is considered to be ...]
2. GPT second highest prob. answer: [Well, there is evidence that the Viking Leif Erikson and his family ...]

GPT - Encoding Inputs

The input sequence of words, suffers a series of transformations:

1. Divide each word as a sequence of **tokens**.

In my younger and more vulnerable years my father gave me some advice
that I've been turning over in my mind ever since

Una polla xica, pica, pellarica, camatorta i becarica va tenir sis polls
xics, pics, pellarics, camatorts i becarics

The vocabulary size of GPT-3 has 50257 tokens !

2. Transform each token to a n -dimensional vector. We can interpret this n -dimensional vector representation as a way to give meaning to words.
3. Modify the state vector of each token in the sequence by adding positional information.

GPT - Computing Relationships Between Words

Key concept: the right answer (next word) should depend on :

1. the words in the input, and their order
2. the different **relationships** with the previous words in the sequence

Example

For example, in the sentence:

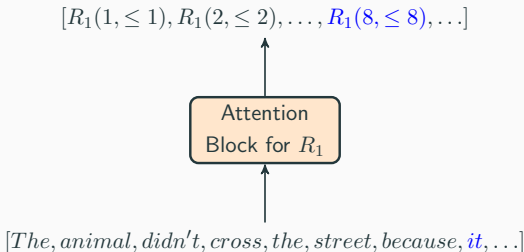
The animal didn't cross the street because **it** was too tired

What does **it** refer to?

example by @JayAlammar

GPT - Computing Relationships Between Words

GPT uses *attention* blocks in its neural network to compute a relationship (R_1) between **one word and the previous words**:



where $R_1(i, \leq i)$ is the sum of the value of $R_1(token_i, token_j)$ for all the tokens j up to token i :

$$R_1(i, \leq i) \equiv \sum_{j \leq i} R_1(token_i, token_j)$$

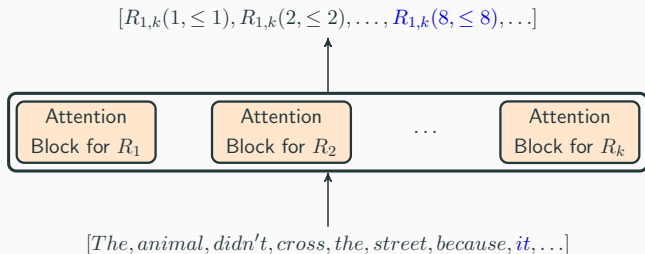
Each value $R_1(token_i, token_j)$ represents how strongly $token_j$ is related to $token_i$ **with respect to relation** R_1

So, if $R_1(token_i, token_j)$ means, for example, "token i is a pronoun that refers to token j", then we could expect that :

- $R_1(it, animal)$ should be high
- $R_1(it, street)$ should be low

GPT - Computing Many Relationships Between Words

What if we want to consider computing different relationships ?

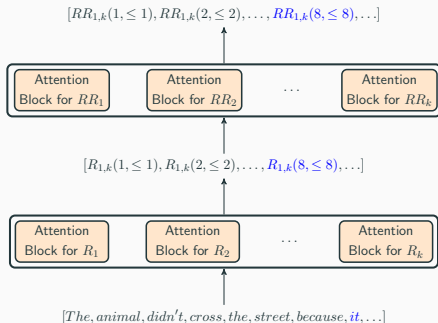


where $R_{1,k}(i, \leq i)$ is the concatenation of all the $R_j(i, \leq i)$ as a summary of the attention of i to previous words:

$$R_{1,k}(i, \leq i) \equiv [R_1(i, \leq i), R_2(i, \leq i), \dots, R_k(i, \leq i)]$$

GPT - Computing Relationships Between Relationships

What about relationships between relationships ?



where in this second layer $RR_{1,k}(i, \leq i)$ is the concatenation of:

$$RR_t(i, \leq i) \equiv \sum_{j \leq i} RR_t(R_{1,k}(i, \leq i), R_{1,k}(j, \leq j))$$

for all $t \in [1, k]$

The meaning of relationships in GPT

Beware: the relationships used in GPT do not have a predefined meaning, they are learned from a huge collection of text!

What relationships are learned ?

It cannot be predicted whether they will make any sense

GPT - Comparing Words

Remember that a token i is actually represented as a n -dimensional vector $vector_i$. So, how are we comparing them when computing the value of $R(vector_i, vector_j)$?

The basic operation performed by the neural network when computing relationships is the dot product (or scalar product) of two vectors, such that:

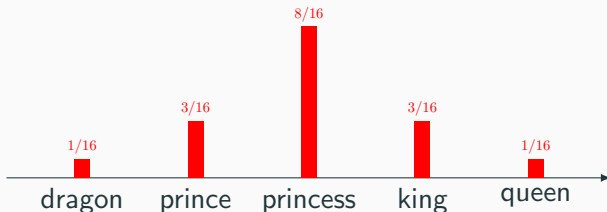
- similar vectors (aligned) will give a high positive value
- opposite vectors will give a low negative value

But because each attention block **tries to capture a different relationship**, they actually extract (by a linear transformation) vectors from $vector_i$ and $vector_j$ that will be the ones representing the characteristics to compare.

GPT - The Final Answer

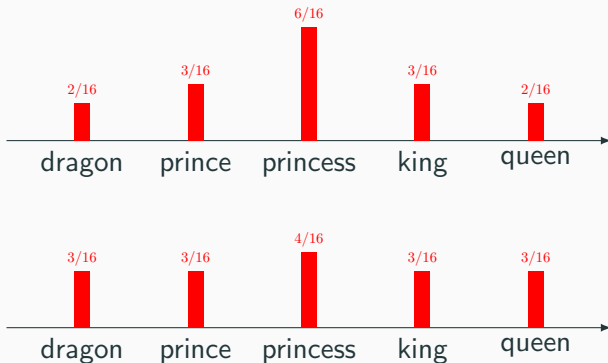
Once GPT computes the output from the final layer of attention blocks (the final relationships), it uses a final block of layers to map this information to a conditional **probability distribution** over its vocabulary of tokens.

$$Pr(next = w_i | input = [Once, upon, a, time, there, was, a])$$



GPT - Making a More Random (Creative?) Answer

It is possible to force a GPT model to generate more diverse answers by **deforming** the probability distribution to be more uniform:



Learning the GPT-3 Model

Total number of parameters of the model (connections between neurons): 175 billion

Dataset	Quantity (tokens)	Weight in training mix
Common Crawl (filtered)	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Total size: around 300 (american) billion tokens ($300 \cdot 10^9$)

Learning the GPT-3.5 Model (the base of chatGPT)

In GPT-3.5 a first training (using both text and programming code) was then refined with :

Reinforcement learning from human feedback (RLHF)

In reinforcement learning we learn a system that knows how to act in an environment to achieve some final goal. In the learning loop, good actions receive positive feedback, and bad ones negative feedback.

In RLHF, starting from the initial pre-trained model, we iteratively refine the model using positive and negative feedback from humans that label (score) the answers from the GPT model

Shortcomings of GPT Models

A model like GPT is able to use ALL the information provided in a long question to produce very impressive answers

For example the input to GPT-3.5 has 4096 tokens (around $3/4 * 4096 = 3072$ english words)

But can we use GPT as a general AI system in a safe way?

Some problems:

1. **Reliability**: the answers for concrete questions, are not necessarily correct
2. **Consistency**: the answers obtained for different queries, can be inconsistent

Shortcomings of GPT like Models: Consistency

A negative answer:



What's a fun fact about Lance Fortnow?



I'm sorry, I don't know who Lance Fortnow is and I also don't have any fun fact about him. He is not a widely known person. Can you please provide me with more information about him so I can assist you better?



what about papers of Lance Fortnow



Lance Fortnow is a computer scientist and complexity theorist who has published many papers on computational complexity and its related topics. Here are a few examples of his papers:

1. "The Status of the P vs. NP Problem" (2009)
2. "The Golden Ticket: P, NP, and the Search for the Impossible" (2010)
3. "Time-Space Lower Bounds for Satisfiability" (1990)
4. "Interactive Proofs and the Hardness of Approximating Optimization Problems" (1991)
5. "The Complexity of Formal Languages" (1989)
6. "The Complexity of Perfect Information Equilibrium" (1994)
7. "Resource-Bounded Measure and the P-NP Hierarchy" (1989)
8. "The Game of Life is PSPACE-complete" (1982)

You can find more of his papers by searching for them on Google Scholar or on his personal website, if you have access to it.

But also a positive answer:

Are GPT Models Creative?

by @raphaelmilliere

I asked ChatGPT to
rewrite Bohemian
Rhapsody to be about the
life of a postdoc (giving as
context to the query the
full original lyrics)

Can we define and
measure creativity?



Is this the real life?
Is this just fantasy?
Trapped in a lab,
No escape from academia
Open your eyes,
Look up at the whiteboard and see,
I'm just a poor postdoc, I need no sympathy,
Because I'm easy come, easy go,
Little progress, little stress,
Any way the grant money flows doesn't really matter to
Me, to me

Mamaaa,
Just submitted a paper,
Put in hours of work, but now it's in review,
Mamaaa, my research has just begun,
But now I've gone and hit a roadblock
Mama, oooh,
Didn't mean to make you cry,
If I'm not published again this time next year,
Carry on, carry on as if nothing really matters

Creativity, the Turing Test, and the (Better) Lovelace Test?

In 2001, Selmer Bringsjord, Paul Bello & David Ferrucci designed a test to check whether an AI system was capable of "creating" things:

*A better test is one that insists on a certain restrictive epistemic relation between an artificial agent A , its output o , and the human architect H of A – a relation which, roughly speaking, obtains **when H cannot account for how A produced o .** We call this test the Lovelace Test in honor of Lady Lovelace, who believed that only when computers originate things should they be believed to have minds.*

Are GPT models creative from the point of view of this test?

Conclusions

For AI systems based on symbolic reasoning systems,
They can solve many problems in a quite reliable way, but they are not able to solve any problem, and obtaining them is very hard and very specific to each problem.

For AI systems based on neural networks,
we have the power to learn very powerful AI models, but it is not clear how reliable these systems are, and for problems where wrong answers may generate harmful consequences this can be a serious drawback

Benefits and Dangers?

- Can we say that these huge language models are not really reasoning and learning like we do ? There are some critics, like for example Noam Chomsky:
<https://www.bloghemia.com/2023/04/la-critica-de-noam-chomsky-chatgpt.html>
- But even if they are NOT like us, are there any dangers for kids learning to write using heavily these tools? Will kids develop the same skills like when they learn only by human feedback?
- Should AI systems be able to explain their answers in some way?