

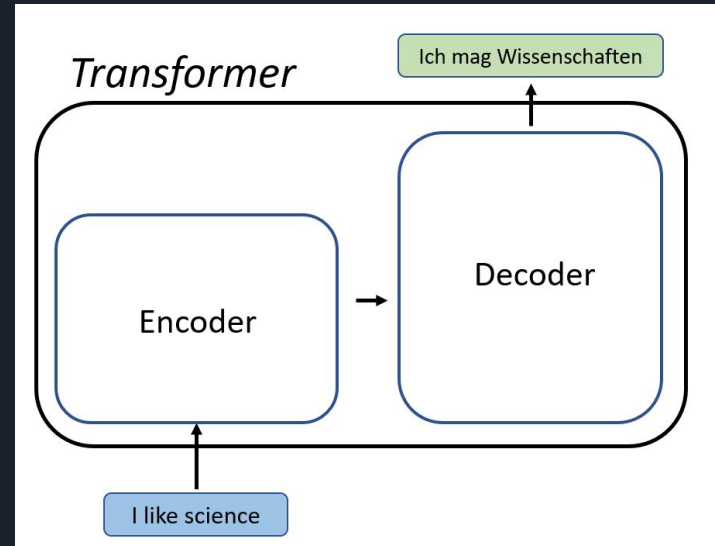
A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green. They are positioned diagonally, with the blue one partially covering the green one.


Transformers, how do they work?

Generative AI to create content

Introduction

Created by Google in 2017 and have a system of attention. This gives different weights for the significance of the training data.





Generative AI is a just a prediction machine

Inference, or predicting the next word based on the previous word is at the heart of generative AI. Compared to AI 20+ years ago, machines are faster and more powerful so they can train models that are magnitudes bigger than former models.

Let's run through an example



Given a simple sentence

Lets say you have a sentence like

“To be, or not to be, that is the question: Whether 'tis nobler in the mind to suffer”

- Shakespeare



We encode it using n grams

N grams is just a grouping of words with “n” elements. N in this case can be anything, lets say 3 and are commonly referred to as tokens.

In the previous sentence:

“To be, or not to be, that is the question: Whether 'tis nobler in the mind to suffer”

- Shakespeare

Lets break this apart into 3-Grams



N Gram conversion

“To be, or not to be, that is the question: Whether 'tis nobler in the mind to suffer”

Becomes

(to, be, or), (be, or, not), (or, not, to), (not, to, be), (to, be, that), ...

This is after stripping the punctuation and turning everything into a lowercase character



Lets build the inference

Now that we have the 3-grams generated, then we can create an inference matrix stating what the next words can be for each n-gram

(to, be, or) -> not


(be, or, not) -> to

(or, not, to) -> be

(not, to, be) -> that

(to, be, that) -> is

So on and so forth...



What happens next depends on what happened in the past

We take a look at the existing input and then infer what the next word will be depending on the existing input. This is why we must give a prompt, this acts as a seed and will start the inference cycle.

Lets say we give a prompt and we're looking at 3 tokens:

To be or not

The input to the LLM is now (be, or, not)



Now lets infer...

Based on the input 3-gram (*be, or, not*) we will need to infer what the next word will be.

Looking at the previous inference matrix, we can infer that if we have an 3-gram of (*be, or, not*) then the next logical word will be that

So now we have a prompt that is:

To be or not to

Now we feed the last 3-gram into the LLM again and get the next predicted word

(*or, not, to*) which infers *be*

This leads to a newly generated prompt of: *To be or not to be*



Hallucinations happen to everyone

The previous example is very simple, but let's give another one that's trickier.

Robert has a white dog, Kathy has a white cat.

Going through the previous examples, let's generate the 3-grams

(Robert, has, a), (has, a, white), (a, white, dog), (white, dog, Kathy), (dog, Kathy, has), (Kathy, has, a), (has, a, white), (a, white, cat)



Lets build the inference

The previous 3-gram conversion becomes

(Robert, has, a) -> white

(has, a, white) -> cat, dog

(a, white, dog) -> Kathy

(white, dog, Kathy) -> has

(Kathy, has, a) -> white

(a, white, cat) -> [end of sentence]



Theres a problem, can you see it?

But theres a fatal problem! Lets prompt the LLM with the prompt,

Robert has a

From the previous 3-grams we know that this 3 gram only has one option afterwards which is white

Now the prompt is *Robert has a white*

Feeding the last 3 gram back in (*has, a, white*) we now have 2 options to pick from, cat or dog.

What do we do?



We flip a coin

Since cat and dog are both equally likely to be picked, we will randomly flip a coin and pick the next word. This is why sometimes LLMs hallucinate.

Lets say we pick dog,

Robert has a white dog, this is correct

However lets say we pick cat

Robert has a white cat, this is not correct



LLMs hallucinate because output is not deterministic

LLMs generally pick the next word based on the existing input. When phrasing is common, then tend to hallucinate since there are more options for next words.