An Introduction to (Large) Language Models

COMS 4774

Aditya Krishna Menon

November 2024

Introduction

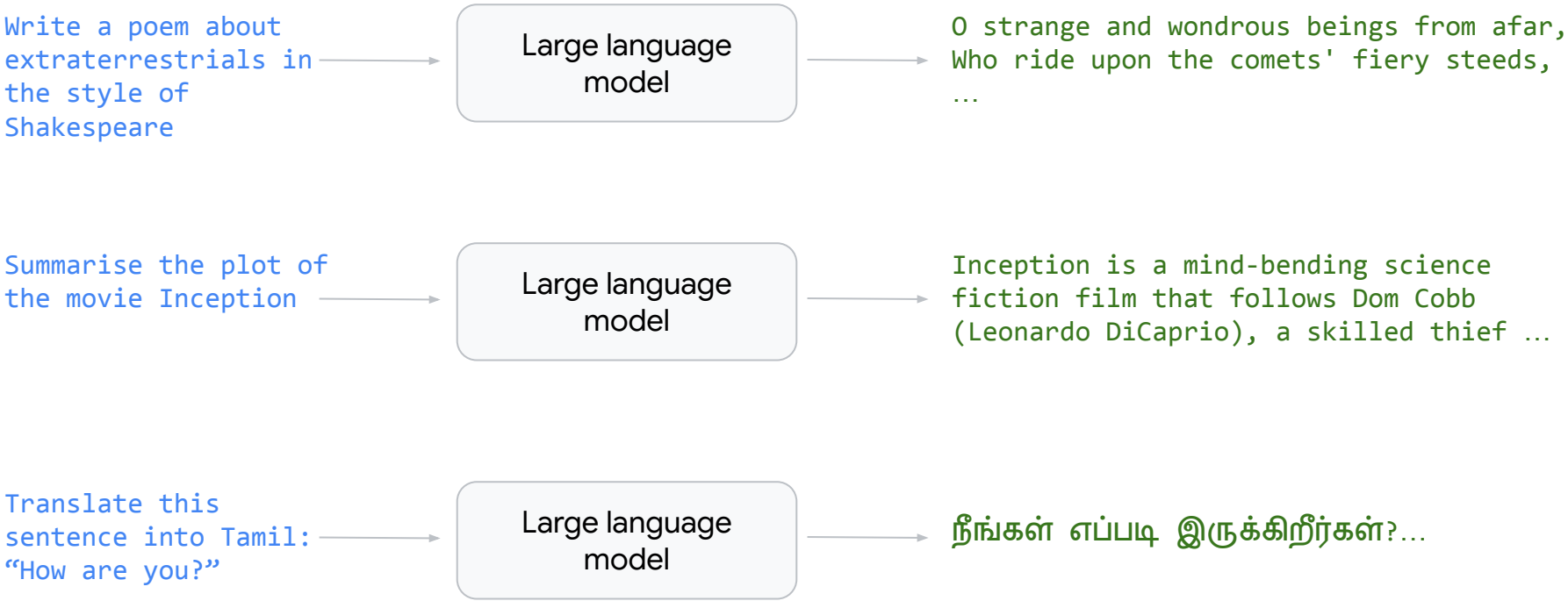
Research Scientist at Google NYC

Working on retrieval models & language model efficiency

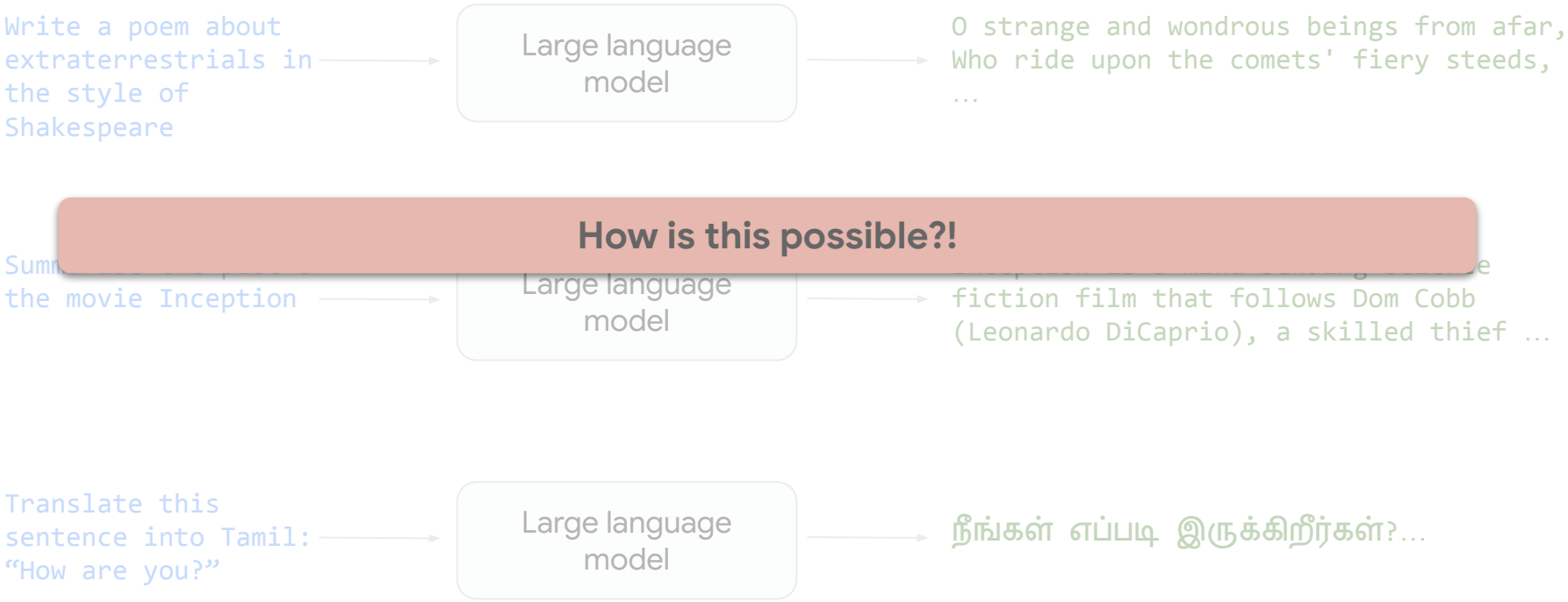
Past lives:

- University of Sydney
- University of California San Diego
- NICTA / CSIRO Data61 / ANU

Motivating question



Motivating question



Motivating sub-questions

→ Large language model → ???

Write a poem about extraterrestrials in the style of Shakespeare → Large language model → is what my professor once told me. They weren't joking, they said my final grade depended on it...

Write a poem about extraterrestrials in the style of Shakespeare → Large language model → No. Why don't *you* try writing a poem?

🤔 How to model the inherent structure of language?

Motivating sub-questions

→ Large language model → ???

Write a poem about extraterrestrials in the style of Shakespeare → Large language model → is what my professor once told me. They weren't joking, they said my final grade depended on it...

Write a poem about extraterrestrials in the style of Shakespeare → Large language model → No. Why don't *you* try writing a poem?

🤔 How to follow user-provided instructions?

Motivating sub-questions

→ Large language model → ???

Write a poem about extraterrestrials in the style of Shakespeare

→ Large language model → is what my professor once told me. They weren't joking, they said my final grade depended on it...

Write a poem about extraterrestrials in the style of Shakespeare

→ Large language model → No. Why don't *you* try writing a poem?

🤔 How to align to user preferences?

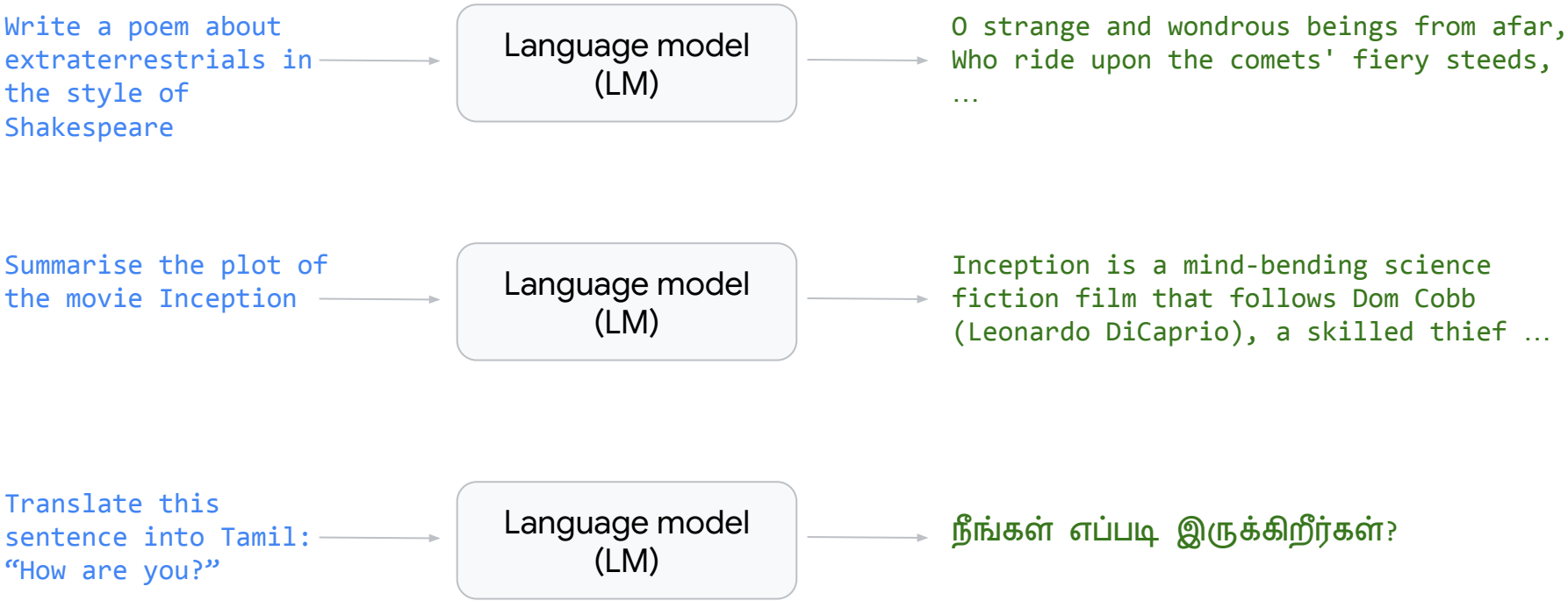
📚 (More) Motivating sub-questions

Rich, active area of research!

- Pre-training data (mixture) selection
- Faster inference mechanisms
- Transformer variants and alternatives
- Retrieval-augmented generation
- Representation power
- Scaling laws
- Emergent abilities
- Reasoning
- ...

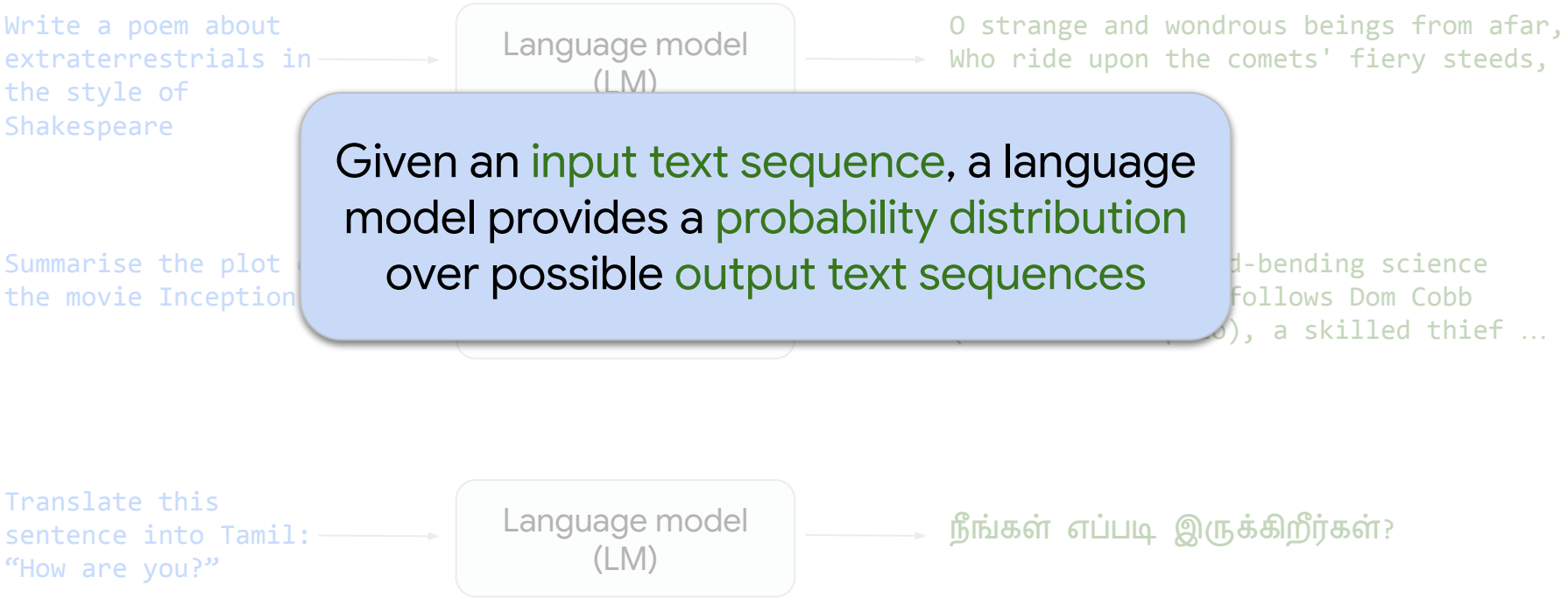
Language models: a bird’s eye view

What is a language model (LM)?



Technically, this is sometimes explicitly referred to as a *conditional* language model.

What is a language model (LM)?



Given an input text sequence, a language model provides a probability distribution over possible output text sequences

Technically, this is sometimes explicitly referred to as a *conditional* language model.

Probability distribution: example

Input	Candidate output	LM log-probability
Write a poem about extraterrestrials in the style of Shakespeare	O strange and wondrous beings from afar, Who ride upon the comets' fiery steeds, …	-2.0
	Alien, alien, You're not mammalian …	-5.0
	'Tis a sight to see, the extraterrestrial. See how they cross galaxies untold, in search of life …	-6.0
	I like eggs	-600.0

Smaller numbers → less likely

Language model: formally

Let V be a finite, non-empty vocabulary of tokens

e.g., { a, b, …, z, 0, 1, …, 9, ␣ }[1]

Let V^* denote the set of all finite-length sequences generated by V

i.e., $V^* = \{\varepsilon\} \cup V \cup V^2 \cup V^3 \cup \ldots$

Classically, we define:

A language model is a probability distribution $p_{LM}(\cdot)$ over V^*

[1] In practice, the vocabulary may comprise (sub)words, and cover multiple languages

Conditional language models

We will make two further amendments

First, we are interested in settings where there is a context $x \in V^*$

Second, let $V_{val}^* \subset (V \cup \{ \$ \})^*$ denote all "valid" sequences containing a single terminal symbol $\$ \notin V$ at (and only at) the final position

Now, we define:

A (conditional) language model is a family of probability distributions $\{ p_{LM}(\cdot \mid x) : x \in V^* \}$, where each distribution is over V_{val}^*

See Formal Aspects of Language Modeling, Cotterell et al., for a more careful exposition.

Language model: formally

Let V be a finite, non-empty vocabulary of tokens

e.g., { a, b, …, z, 0, 1, …, 9, ␣ }[1]

Let V^* denote the set of all finite-length sequences generated by V

i.e., $V^* = \{\varepsilon\} \cup V \cup V^2 \cup V^3 \cup \ldots$

Let $ denote a special terminal token

We assume that $\$ \notin V$

Let $V_{val}^* \subset (V \cup \{ \$ \})^*$ denote all "valid" sequences containing a single $ at (and only at) the final position

[1] In practice, the vocabulary may comprise (sub)words, and cover multiple languages

Language model: formally

Given a context $x \in V^*$, a (conditional) language model is a probability distribution $p_{LM}(\cdot \mid x)$ over sequences in V_{val}^*

We generally consider a family of models $\{ p_{LM}(\cdot \mid x) : x \in V^* \}$

Some technical subtleties; see later!

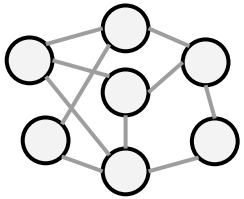
The context $x \in V^*$ could be the empty string ε

Sometimes referred to as unconditional language model

Image classification pipeline

Training

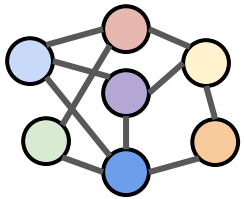
Initial model
(e.g., neural network
with random weights)



Input data
("training" set)

Training procedure

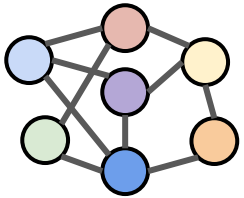
Trained model
(e.g., neural network
with optimised weights)



Inference

New input
("test" example)

Trained model



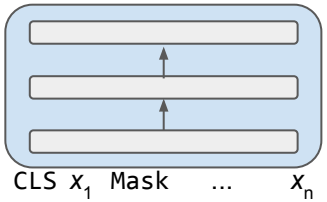
Predicted label

0.3	0.1	0.5	0.2
Parrot	Magpie	Cockatoo	Pigeon

Language modelling pipeline

Training

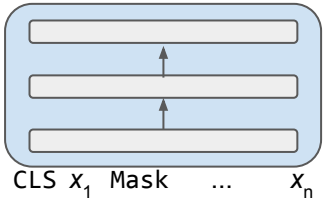
Initial model (e.g., Transformer with random weights)



Input data ("training" set)

Training procedure

Trained model (e.g., Transformer with optimised weights)

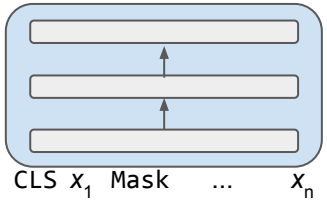


Inference

New input ("test" example)

```
Summarise the plot
of the movie
Inception
```

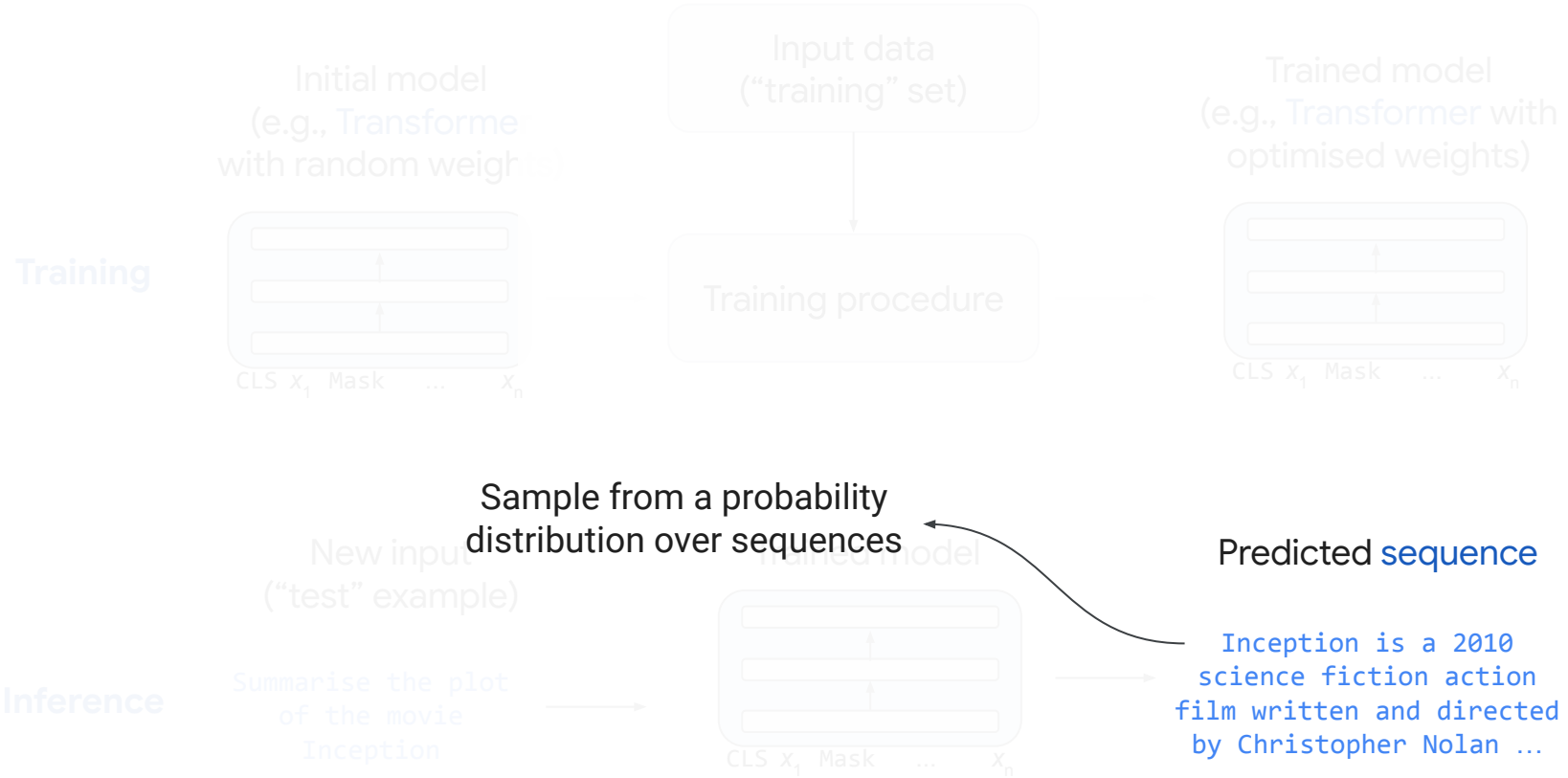
Trained model



Predicted sequence

```
Inception is a 2010
science fiction action
film written and directed
by Christopher Nolan ...
```

Language modelling pipeline



Probabilistic next-token prediction

Probabilistic model over sequences

Let $x = x_1\ x_2\ \ldots\ x_m$ denote a context sequence of m tokens

e.g., x_1 = "`Write`", x_2 = "`a`", x_3 = "`poem`", ...

According to our definition, an LM produces a distribution

$$p_{LM}(y \mid x)$$

over possible (valid) output sequences $y = y_1\ y_2\ \ldots\ y_n\$$

Needs to work for any m, n — not known a-priori!

Typically, $m \neq n$

Probabilistic model over sequences

Let $x = x_1\ x_2\ \ldots\ x_m$ denote a context sequence of m tokens

e.g., x_1 = "`Write`", x_2 = "`a`", x_3 = "`poem`", ...

According to $p_{LM}(y \mid x)$

😟 It's hard to write down $p_{LM}(y \mid x)$
Set of possible y is very large!

over possible (valid) output sequences $y = y_1\ y_2\ \ldots\ y_n\$$

Needs to work for any m, n — not known a-priori!

Typically, $m \neq n$

Conditional probability decomposition

Invoking the chain rule of probability: [⚠️ see next slide ⚠️]

$$p_{LM}(y \mid x) = p_{LM}(y_1 \mid x) \cdot p_{LM}(y_2 \mid x, y_1) \cdot p_{LM}(y_3 \mid x, y_1, y_2) \ldots$$
$$= \prod_i p_{LM}(y_i \mid x, y_{<i}) \cdot p_{LM}(\$ \mid x, y)$$

where $y_{<i} = y_1\, y_2 \ldots y_{i-1}$

Thus, we can sample tokens one at a time!

🤓 Our LM family defines $p_{LM}(\cdot \mid x)$ for **any** x, y, so $p_{LM}(y_i \mid x, y_{<i})$ is well defined

📚 Language models and processes

Consider a stochastic process $(X_t)_{t \in N}$, with X_t taking values in $V \cup \{ \$ \}$

- Call this a “language process”
- The chain rule may be applied to this process

We can associate any family $\{ p_{LM}(\cdot \mid x) : x \in V^* \}$ with a language process

- Kolmogorov extension theorem

However, some families $\{ p_{LM}(\cdot \mid x) : x \in V^* \}$ may yield infinite sequences!

- Need to ensure “tightness”
- Sufficient condition: $p_{LM}(\$ \mid x)$ doesn’t decay too fast with length of x

See When is a Language Process a Language Model?, ACL 2024

Next-token distribution modelling: second attempt

🤓 **If** we can model each $p_{LM}(y_i \mid x, y_{<i})$, we can sample sequences!

🤔 **How** do we model $p_{LM}(y_i \mid x, y_{<i})$?

Modest first step: rewrite it as

$$p_{LM}(y_i \mid x, y_{<i}) = \exp(s(x, y_{<i}, y_i)) \,/\, \Sigma_{y'} \exp(s(x, y_{<i}, y'))$$

Here, $s(x, y_{<i}, y')$ measures how well y' completes the sequence $x, y_{<i}$

Contextual score modelling

🤔 How do we model $s(x, y_{<i}, y')$?

Simple choice is a linear model:

$$s(x, y_{<i}, y') = w_{y'}^{T} \Phi(x, y_{<i})$$

cf. multi-class logistic regression!

Each token $t \in V \cup \{ \$ \}$ is embedded into a vector $w_t \in R^D$

Each sequence $z \in (V \cup \{ \$ \})^*$ is embedded into a vector $\Phi(z) \in R^D$

Sequence embedding

🤔 How do we parameterise $\Phi(x, y_{<i})$?

The sequence embedder $\Phi(x, y_{<i})$ must work for *any* sequence length!

💡 Average the individual token embeddings!

Next-token distribution modelling: summary

🤔 How do we model $p_{LM}(y_i \mid x, y_{<i})$?

We have proposed:

$$p_{LM}(y_i \mid x, y_{<i}) = \exp(s(x, y_{<i}, y_i)) \,/\, \Sigma_{y'} \exp(s(x, y_{<i}, y'))$$

$$s(x, y_{<i}, y_i) = w_{yi}^{T}\, \Phi(x, y_{<i})$$

$$\Phi(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_{i-1}) = (w_{x1} + w_{x2} + \ldots w_{yi-1}) \,/\, (m + i - 1)$$

Next-token distribution modelling: summary

🤔 How do we model $p_{LM}(y_i \mid x, y_{<i})$?

We have proposed:

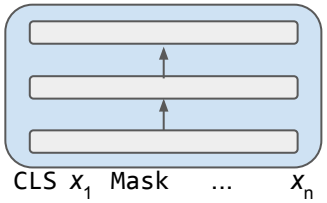
🤔 Can we do better?

$$s(x, y_{<i}, y_i) = w_{yi}^T \Phi(x, y_{<i})$$

$$\Phi(x_1, x_2, ..., x_m, y_1, y_2, ..., y_{i-1}) = (w_{x1} + w_{x2} + ... w_{yi-1}) / (m + i - 1)$$

Language modelling pipeline

Initial model
(e.g., Transformer
with random weights)



Stack of **attention**
and **feed-forward**
layers

Attention & Transformers

When averaging goes bad

Our proposed sequence embedder:

$$\Phi(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_{i-1}) = (w_{x1} + w_{x2} + \ldots w_{yi-1}) / (m + i - 1)$$

This assumes fixed “meanings” of the individual tokens

However, a token’s “meaning” can change with context:

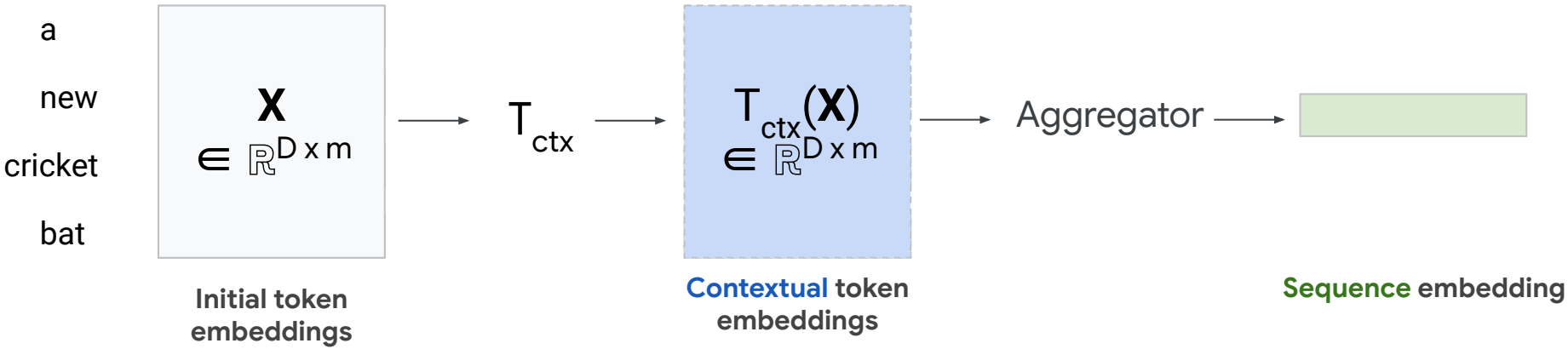
I just got a new cricket bat

Contextual averaging

Ideally, we don't want to rely on static token embeddings

The embedding should vary depending on the context

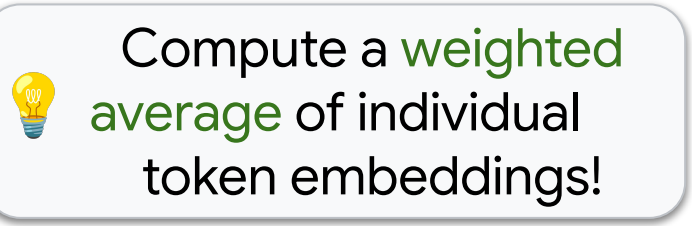
Instead, we want to transform these to contextual token embeddings



Contextual averaging: intuition

Intuitively, embedding of token x_i should be influenced by other tokens x_j

The strength of influence ought to vary across the different tokens



Given the matrix of token embeddings $\mathbf{X} \in \mathbb{R}^{D \times m}$, construct:

$$T_{ctx}(\mathbf{X}) = \mathbf{X}\,\mathbf{A}(\mathbf{X})$$

for suitable weight matrix $A(\mathbf{X}) \in \mathbb{R}^{m \times m}$

Uniform matrix ⟶ standard averaging!

Contextual averaging: instantiation

Construct:

$$T_{ctx}(\mathbf{X}) = \mathbf{X}\,\mathbf{A}(\mathbf{X})$$

Intuitively, $\mathbf{A}(\mathbf{X})_{ij} \in \mathbb{R}$ tells us how much token *i* influences token *j*

A natural thought is to define:

$$\mathbf{A}(\mathbf{X}) = \sigma(\mathbf{X}^T\,\mathbf{X})$$

where σ is column-wise softmax

- Measures the relative similarity between pairs of tokens
- Each column of **A(X)** sums to 1

Contextual averaging: instantiation

Construct:

$$T_{ctx}(\mathbf{X}) = \mathbf{P}\,\mathbf{W}_v^T\,\mathbf{X}\,\mathbf{A}(\mathbf{X})$$

Intuitively, $\mathbf{A}(\mathbf{X})_{ij} \in \mathbb{R}$ tells us how much token *i* influences token *j*

More generally, measure similarity in some **learned projection space**:

$$\mathbf{A}(\mathbf{X}) = \sigma(\mathbf{X}^T\,\mathbf{W}_k^T\,\mathbf{W}_q\,\mathbf{X})$$

where $\mathbf{W}_q \in \mathbb{R}^{k \times D}$, $\mathbf{W}_k \in \mathbb{R}^{k \times D}$, σ is column-wise softmax

Projections allow for **asymmetric** influence!

The attention operator

We have arrived at the attention operator!

$$\texttt{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{V}\,\sigma(\mathbf{K}^T\,\mathbf{Q})$$

for "queries" $\mathbf{Q} \in \mathbb{R}^{k \times m}$, "keys" $\mathbf{K} \in \mathbb{R}^{k \times m}$, "values" $\mathbf{V} \in \mathbb{R}^{k \times d}$

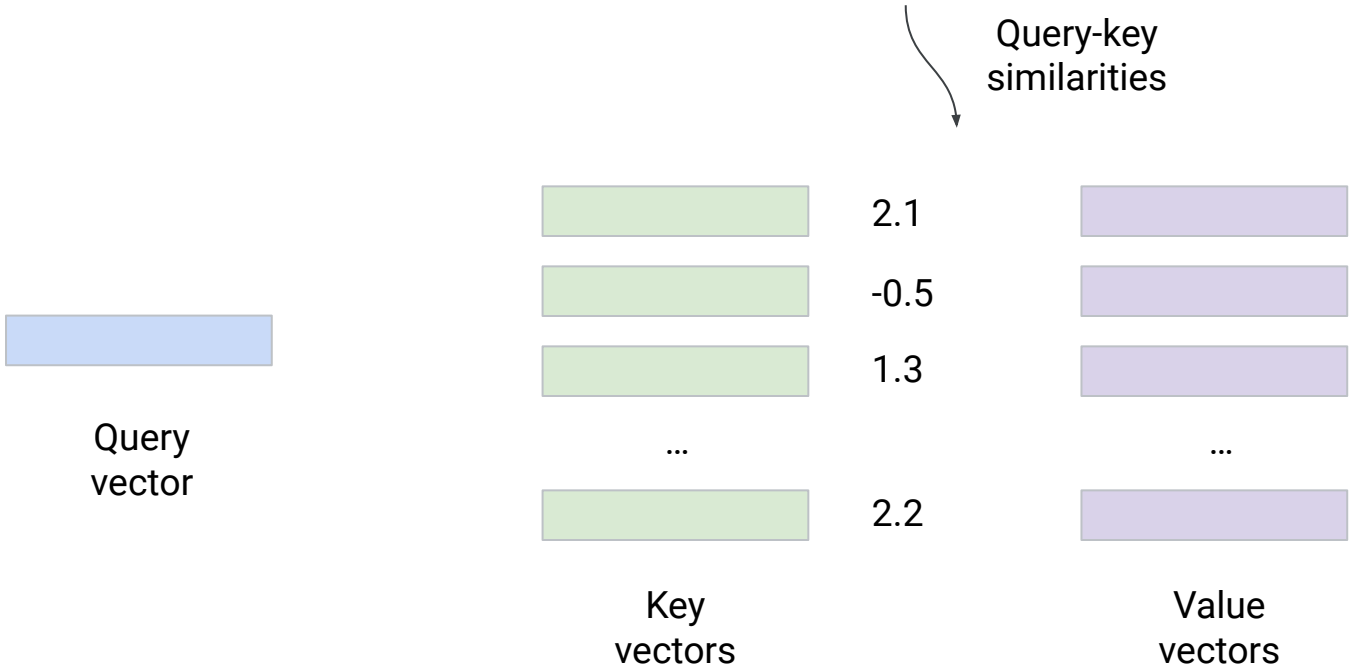
Specifically, we chose

$$T_{ctx}(\mathbf{X}) = \mathbf{P}\,\texttt{Attention}(\mathbf{W}_q^T\mathbf{X}, \mathbf{W}_k^T\mathbf{X}, \mathbf{W}_v^T\mathbf{X})$$

for $\mathbf{P} \in \mathbb{R}^{d \times k}$

Attention as a soft lookup table

$$\text{Attention}(Q, K, V) = V\,\sigma(K^T Q)$$



Attention as a soft lookup table

$$\text{Attention}(Q, K, V) = V\,\sigma(K^T Q)$$

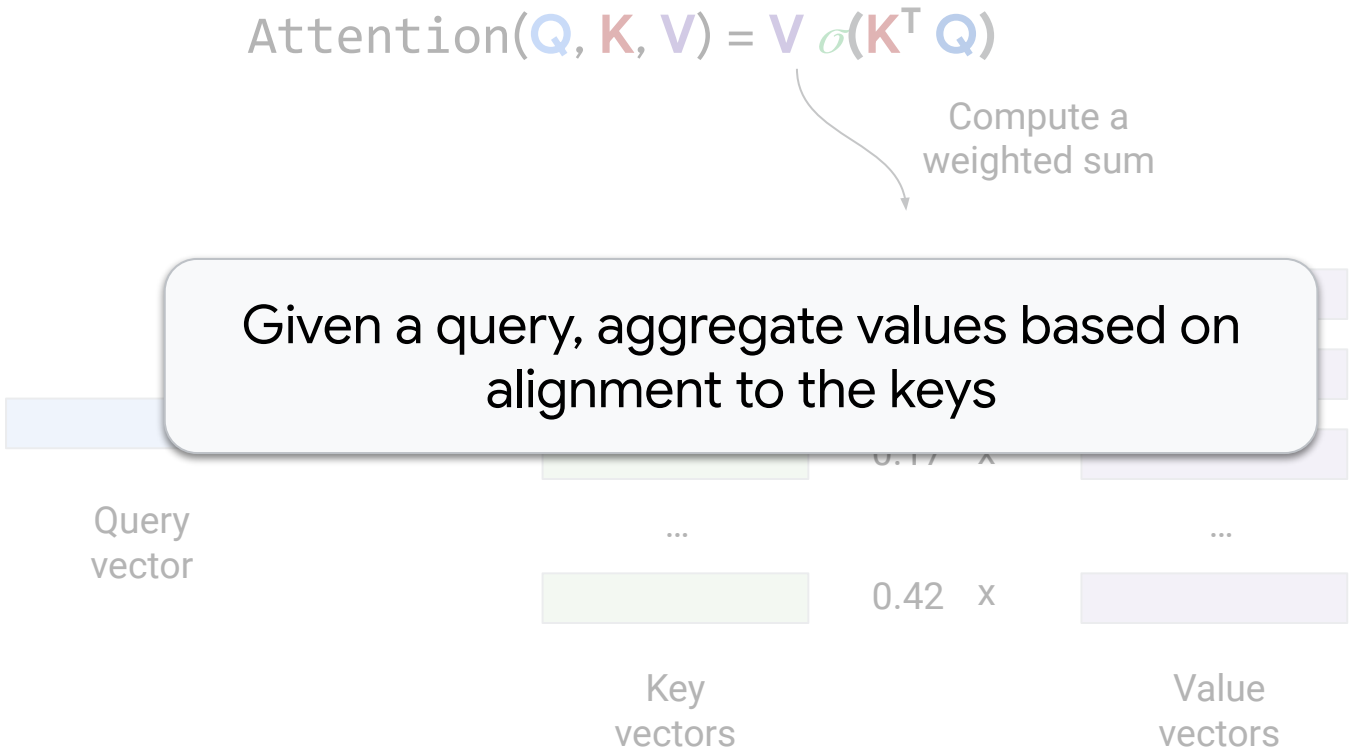
Convert to a probability distribution

Attention as a soft lookup table

$$\text{Attention}(Q, K, V) = V\,\sigma(K^T Q)$$

Compute a weighted sum

Attention as a soft lookup table



Feedforward layers

Two further extensions are useful:

$$T_{ctx}(\mathbf{X}) = FF(Attn(\mathbf{X}))$$

Feedforward layer, providing non-linearity upon stacking

$$FF(\mathbf{Z}) = \mathbf{Z} + \mathbf{C}\,\mathrm{ReLU}(\mathbf{B}\,\mathbf{Z})$$

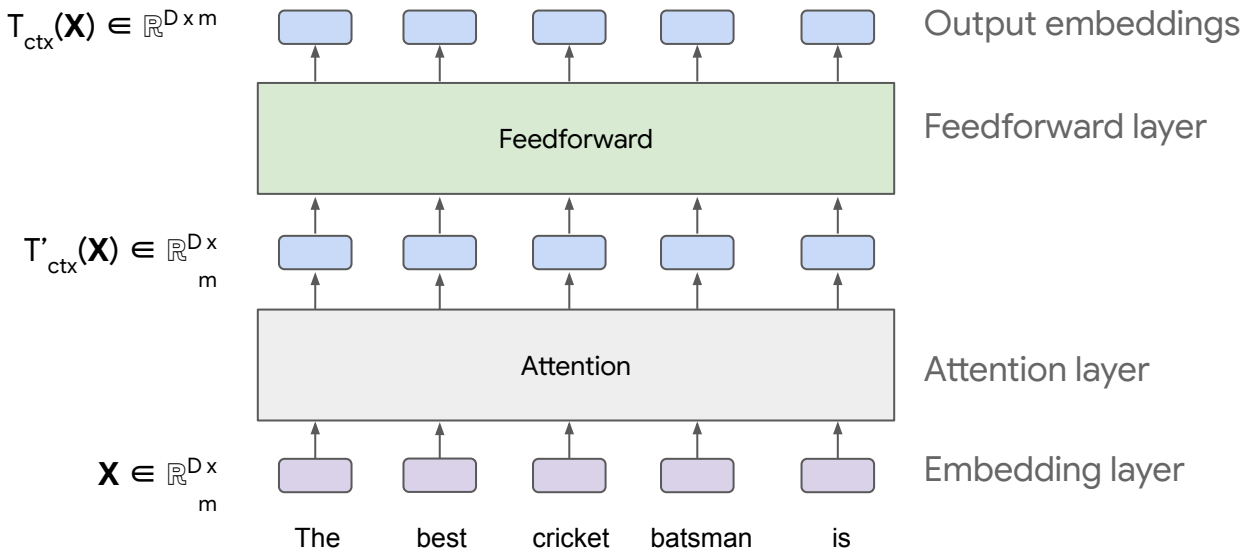
Usually "up-project" into D_h-dimensions

$$Attn(\mathbf{X}) = \mathbf{X} + \mathbf{P}\,\mathtt{Attention}(\mathbf{W}_q^T\mathbf{X}, \mathbf{W}_k^T\mathbf{X}, \mathbf{W}_v^T\mathbf{X})$$

Residual term, providing optimisation stability

From attention to Transformers

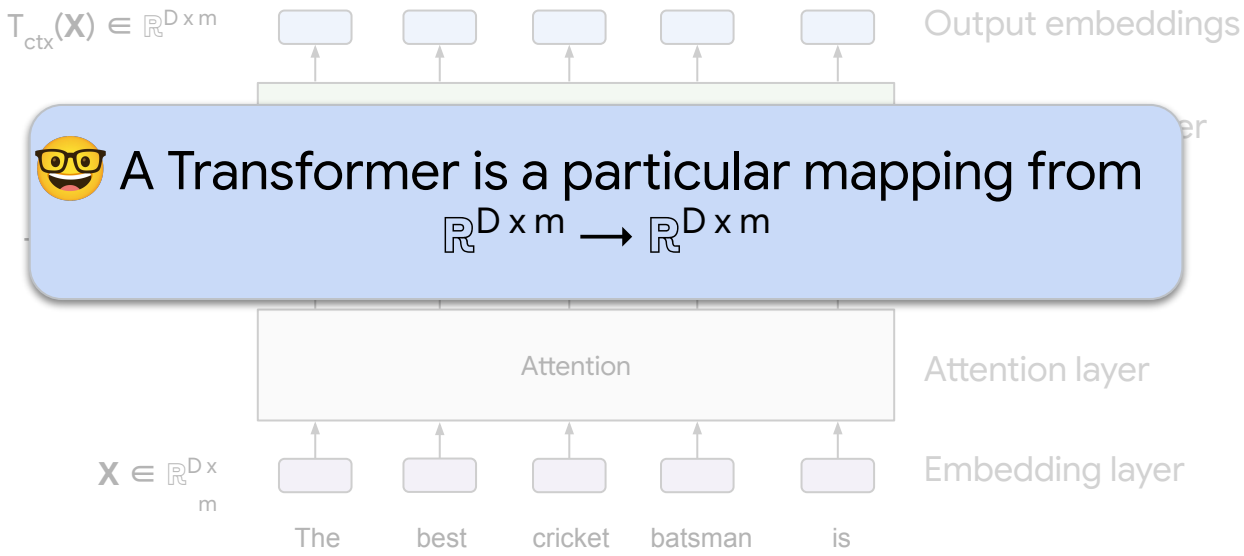
We have arrived at the Transformer model for contextualised embeddings



See "Attention Is All You Need". One typically uses layer normalization as well, but we gloss over this for brevity.

From attention to Transformers

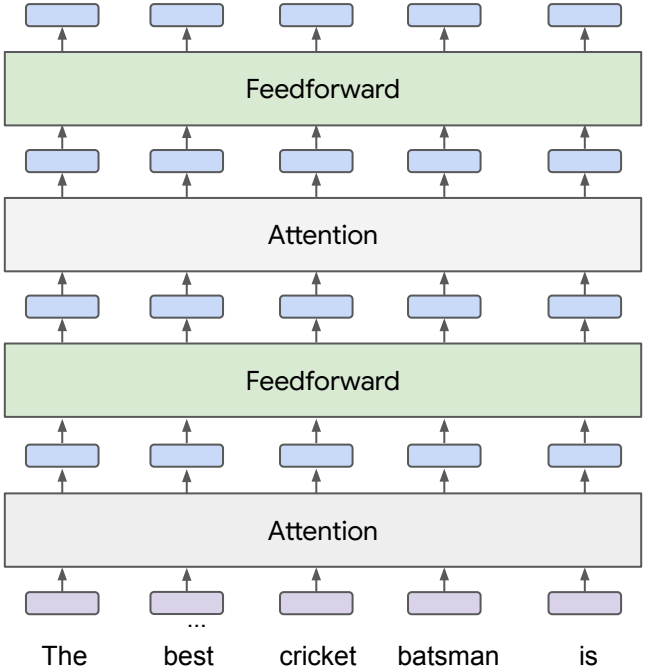
We have arrived at the Transformer model for contextualised embeddings



See "Attention Is All You Need". One typically uses layer normalization as well, but we gloss over this for brevity.

Extensions: multiple layers

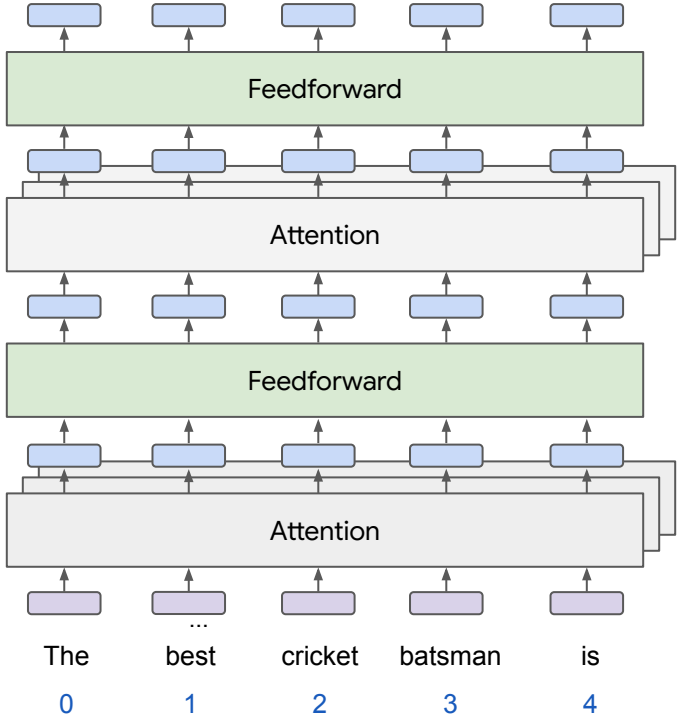
As with standard neural networks, can stack multiple layers



Extensions: position encodings & multi-head attention

Important to also include position encodings & multiple heads

Capture different token “modalities”



📚 Representation power

Transformers are **universal approximators** for sequence-to-sequence functions

i.e., mappings from $\mathbb{R}^{D \times m} \rightarrow \mathbb{R}^{D \times m}$

More precisely, can approximate continuous, permutation-invariant functions with compact support to arbitrary precision $\varepsilon > 0$

Depth $O(m\ (1/\varepsilon)^{Dm} / m!)$ with constant width suffices

Different role of the two layers

Attention layer → implement a contextual mapping of tokens

Feedforward layer → transform contextual mapping to target function

Yun et al., Are Transformers universal approximators of sequence-to-sequence functions?, ICLR 2020

Back to language modelling

Recall that we proposed to use the model:

$$p_{LM}(y_i \mid x, y_{<i}) = \exp(s(x, y_{<i}, y_i)) \,/\, \Sigma_{y'} \exp(s(x, y_{<i}, y'))$$

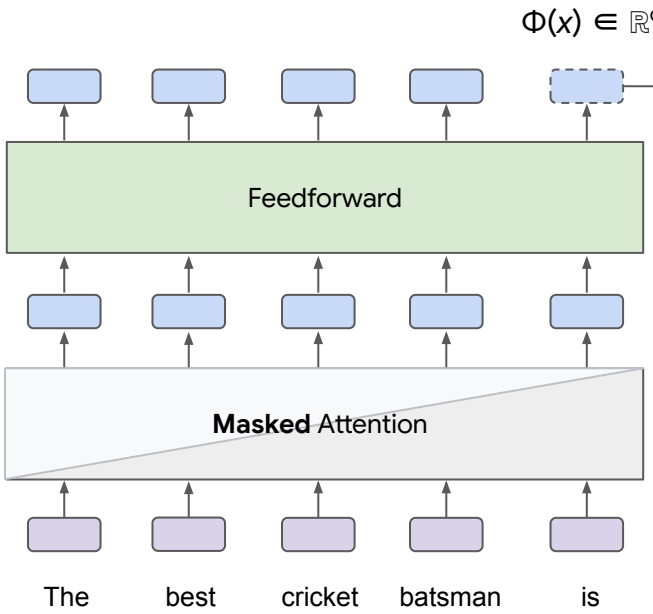
$$s(x, y_{<i}, y') = w_{y'}^{T}\, \Phi(x, y_{<i})$$

How exactly do we get a single embedding $\Phi(x, y_{<i})$ from a Transformer?

Recall this produces a sequence of embeddings for each token

Next-token prediction via Transformers

Simple idea: use the embedding for the final token



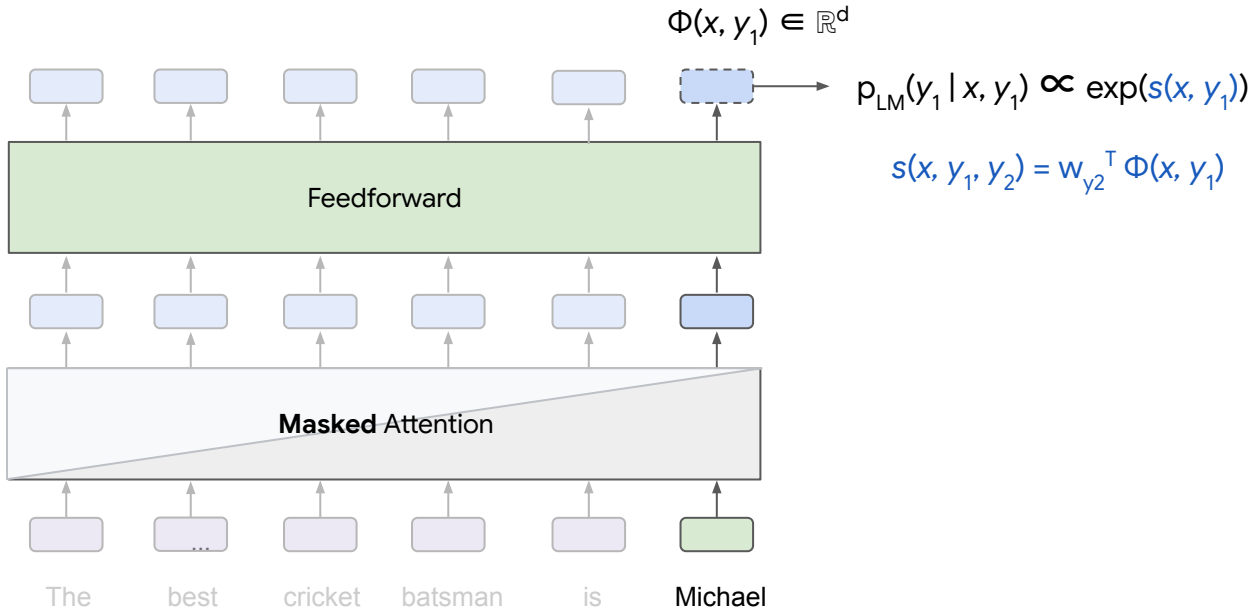
$p_{LM}(y_1 \mid x) \propto \exp(s(x, y_1)) / Z(x)$

$s(x, y_1) = w_{y1}^{T} \Phi(x)$

⚠ Critically requires **masking** the attention matrix; else, we will be peeking into the future!

Next-token prediction via Transformers

We can repeat this process until we reach the terminal symbol ($)!



Summary

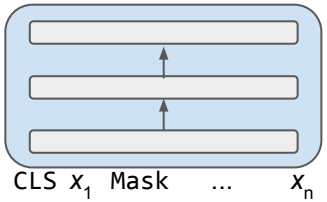
😎 Transformers offer contextual embeddings, which allow for modelling $p_{LM}(y_i \mid x, y_{<i})$

🤔 How do we **fit** the parameters of this model?

Language modelling pipeline

Training

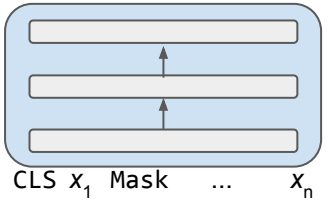
Initial model (e.g., Transformer with random weights)



Input data ("training" set)

Training procedure

Trained model (e.g., Transformer with optimised weights)

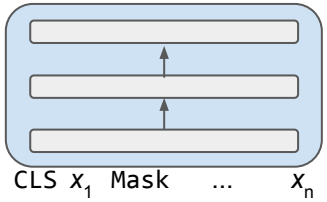


Inference

New input ("test" example)

```
Summarise the plot
of the movie
Inception
```

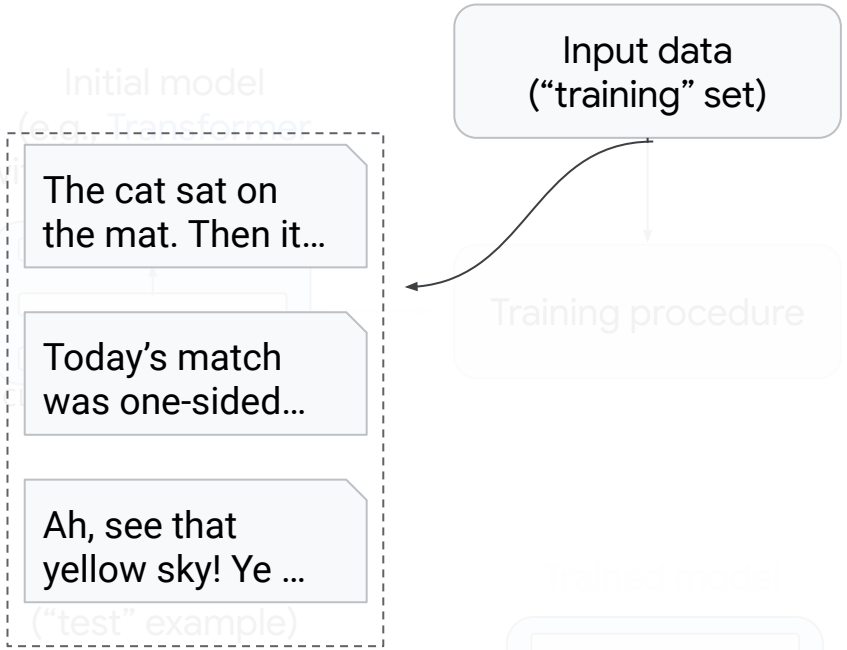
Trained model



Predicted sequence

```
Inception is a 2010
science fiction action
film written and directed
by Christopher Nolan ...
```

Language modelling pipeline



Pre-training, in-context learning, and fine-tuning

Log-likelihood objective

Transformers provide a particular model for $p_{LM}(y_i \mid x, y_{<i})$

This has a number of parameters, e.g., $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, ...

Natural idea: given a (input, output) sequence pair (x, y), minimise

$$-\log p_{LM}(y \mid x) = \Sigma_i -\log p_{LM}(y_i \mid x, y_{<i})$$

i.e., minimise negative log-likelihood

🤔 Where do we get (x, y) from?

Next-token prediction

Say we have a long string of text, e.g., an article

x *y*

Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea...

Next-token prediction

Say we have a long string of text, e.g., an article

x *y*

Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea...

Next-token prediction

Say we have a long string of text, e.g., an article

x x y x y y

Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. I nd regulating the circulat t the mouth; whenever it never I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea...

🤓 Given a sequence, predict the next token

Next-token prediction and pre-training

Next-token prediction is an example of a pre-training objective

Key ingredients:

(1) Large corpus of text

(2) Construction of (x, y) given the corpus

Given (x^j, y^j) pairs constructed this way, minimise

$$\Sigma_j \text{ -log } p_{LM}(y^j \mid x^j)$$

Remarkably effective if we have a large enough set of pairs!

See "Language Models are Unsupervised Multitask Learners"

Does pre-training suffice?

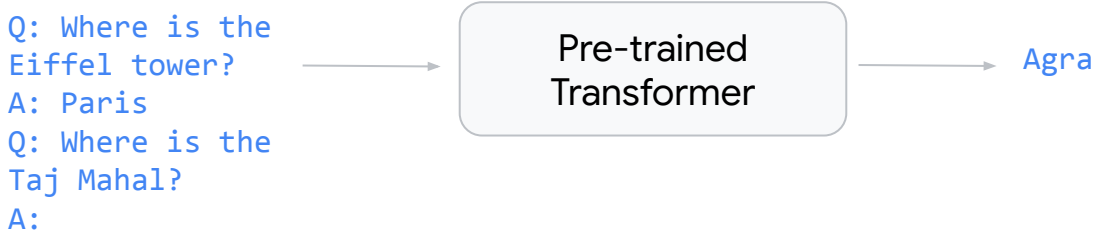
Suppose we have a Transformer pre-trained with next-token prediction

😟 Results may not be coherent!

In-context learning

We can condition the model by choosing the prompt carefully!

In-context learning or few-shot prompting



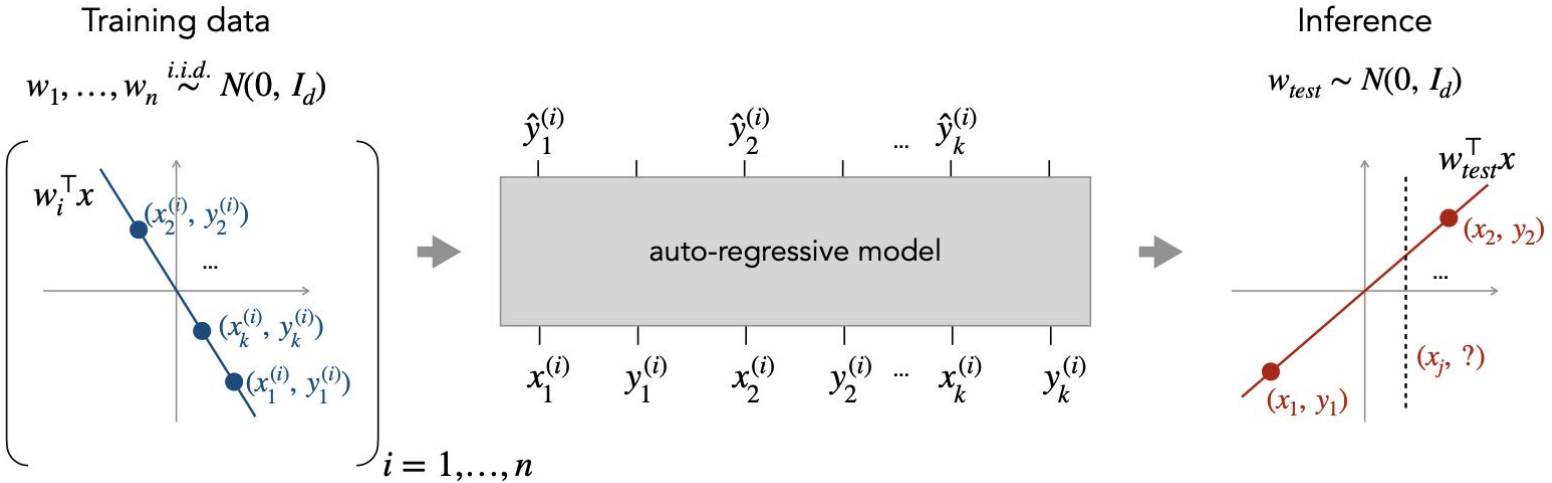
🤓 Condition the model with suitable context, so that $p_{LM}(y \mid x)$ is meaningful

See "Language Models are Few-Shot Learners"

📚 In-context learning

In-context learning offers an intriguing new learning paradigm

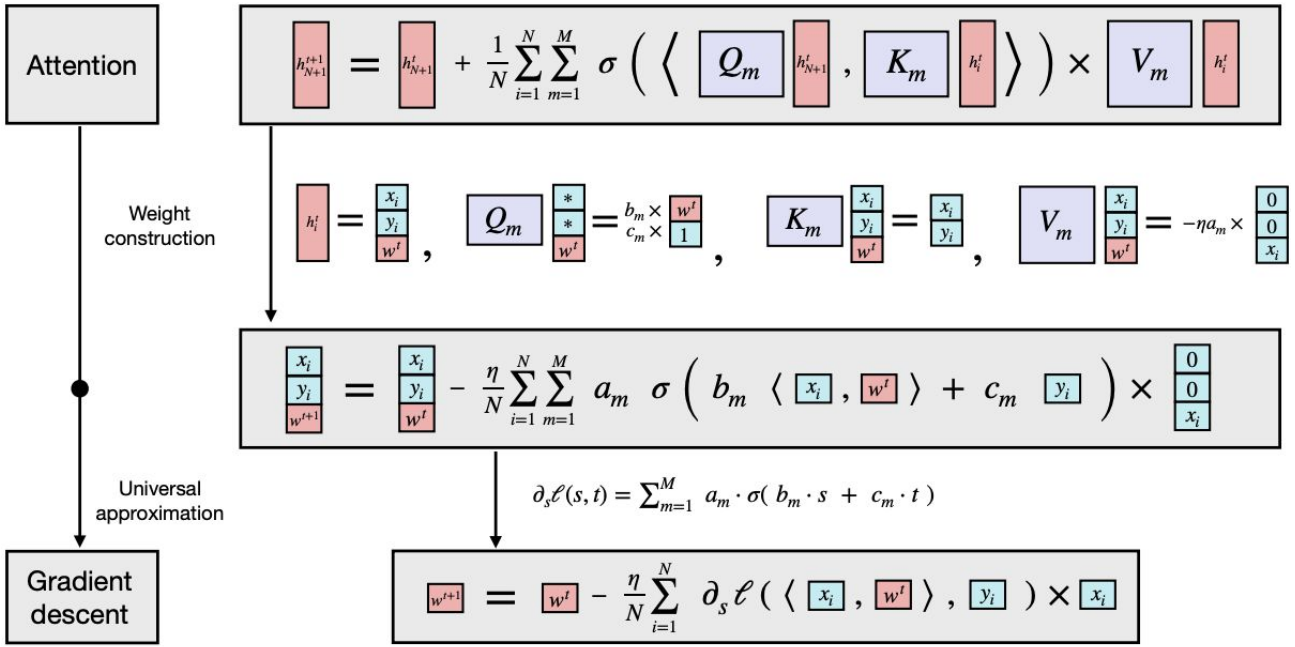
Transformers can learn to solve regression problems, e.g., in-context!



See What Can Transformers Learn In-Context? A Case Study of Simple Function Classes, NeurIPS 2022

In-context learning

Attention layer can mimic gradient descent update!

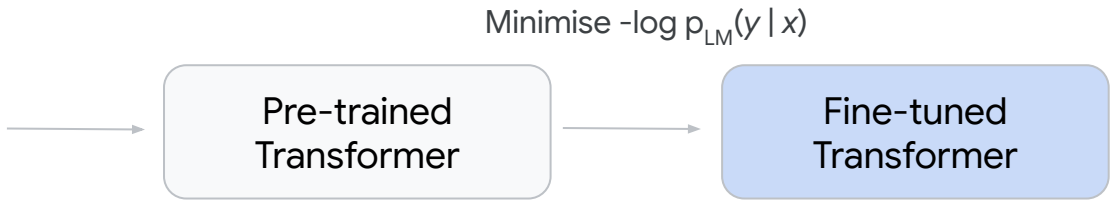


See Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection, NeurIPS 2023

Fine-tuning

Alternately, given many examples, we can fine-tune the model:

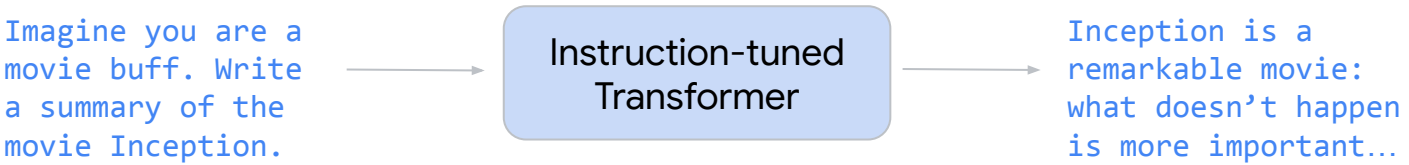
x	*y*
Where is the Eiffel tower?	Paris
When was Rome founded?	753 BC
How many inches in a foot?	12



Instruction tuning

Frame examples from multiple tasks in the form of instructions

x	*y*
Provide a one or two word answer to this question. Where is the Eiffel tower?	Paris
Answer the following with reasoning. If I have 5 apples and give away 2, how many do I have left?	Start with 5 apples. Take away 2 and we have 5 - 2 = 3 apples. So, 3 apples.
Answer yes, no, or indeterminate. Suppose f is convex. Is f concave?	Indeterminate



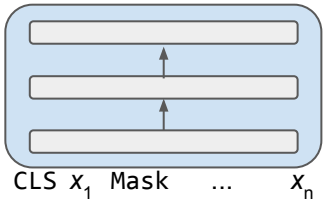
See "Multitask Prompted Training Enables Zero-Shot Task Generalization", "Finetuned Language Models Are Zero-Shot Learners"

Summary

Summary

Training

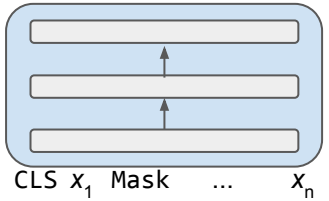
Initial model (e.g., Transformer with random weights)



Input data ("training" set)

Training procedure

Trained model (e.g., Transformer with optimised weights)

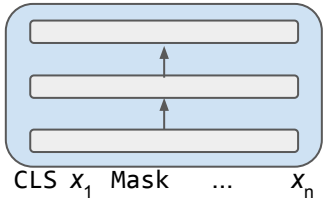


Inference

New input ("test" example)

`Summarise the plot of the movie Inception`

Trained model



Predicted sequence

`Inception is a 2010 science fiction action film written and directed by Christopher Nolan ...`

Further reading

The Illustrated Transformer

Transformers-based Encoder-Decoder Models

Formal Aspects of Language Modeling

Formal Algorithms for Transformers

Language model inference: theory and algorithms

Fundamentals of Transformers

Large Language Models (in 2023)

Further reading

Stanford CS224N

Princeton COS 597R

Harvard IACS CS109

Acknowledgements

Images

Gang-Gang cockatoo

Emojis

Noto Color Emoji