Large, neural probabilistic language models

Mikel L. Forcada[1]

[1]Prompsit Language Engineering, S.L.,
Edifici Quorum III, Av. Universitat s/n, E-03202 Elx

LT-Bridge Spring/Summer School
5 March 2024
(online)

Índex

Index

Language models

A probabilistic language model, or simply a language model is a mathematical device that assigns a probability (likelihood) P to a sequence of words:

$$P(\textit{The chairman of the board}) > P(\textit{The chairman of the cat})$$

$$P(\textit{My cat is black}) > P(\textit{My black is cat})$$

$$P(\textit{My cat is black}) > P(\textit{My cat is green})$$

n-gram models/1

- Language models based on *n-grams* compute the probability P word by word.
- They try to predict the probability of a word after $n-1$ words.
- For example, if $n = 3$ ("trigram" model), probabilities are of the form

$$p_3(\textit{green} \mid \textit{cat is}),$$

that is, the probability of the word *green* after the words *cat is*.

n-gram models/2

A trigram model would compute the probability of *My cat is black* by sliding a window of three words from left to right:

$$
\begin{aligned}
P(\textit{My cat is black .}) = \quad & p_3(\textit{My} \mid --) \times \\
\times \quad & p_3(\textit{cat} \mid -\,\textit{My}) \times \\
\times \quad & p_3(\textit{is} \mid \textit{My cat}) \times \\
\times \quad & p_3(\textit{black} \mid \textit{cat is}) \times \\
\times \quad & p_3(\textit{.} \mid \textit{is black})
\end{aligned}
$$

n-gram models/3

- The probabilities $p_3(is \mid My\ cat)$ are *estimated*.
- The estimate is made by *counting* (*statistical* model) in a corpus with large quantities of text how many times the sequence *My cat* ***word*** occurs for all possible words ***word***.
- Probabilities $p_3(is \mid My\ cat)$ are stored in a *table*.

n-gram models/4

- n-gram models consider only a context of $n-1$ preceding words when it comes to predict the next word.
- We can increase n to have more context, but,
- if the vocabulary has a size V...
- ...we would need a table with

$$V^n = \underbrace{V \times V \times V \times \ldots \times V}_{n \text{ times}}$$

entries.

- If V are 10,000 words (a small vocabulary) and n is a (very small!) context of 4 words ("tetragrams"), we would have $10,000 \times 10,000 \times 10,000 \times 10,000 = 10,000,000,000,000,000$ entries!

n-gram models/5

- 10,000,000,000,000,000 tetragrams would not fit in the memory of most computers.
- Moreover, not all 10,000,000,000,000,000 tetragrams ($n = 4$) appear in a text corpus.
  - For each tetragram to appear at least once, we would need a corpus of at least 10,000,000,000,000,003 words.
- If in a new text a new unseen tetragram appears, we have to estimate the probability using a trigram ($n = 3$);
- if not available, using a bigram ($n = 2$), etc. $\rightarrow$ loss of context $\rightarrow$ worse prediction.

n-gram models/6

Large language models:

- In a large language model, $n > 1,000$, for example, $n = 2{,}049$.[1]
- Clearly, it would not be possible to use a table (it would not fit in the universe!).
- Therefore, it can not be built by just counting events at a corpus and storing their probabilities.
- A mechanism is needed to *compute*

$$p_{2049}(\textbf{\textit{word}} \mid \ldots \textit{(the 2,048 preceding words)} \ldots)$$

[1] That is to say, $2^{11} + 1 = 2{,}048 + 1 = 2{,}049$

Predicting the next word is not easy /1

- Language models learn to *predict the word that follows* a certain number of preceding words.
- This task is not at all easy.
- To succeed, we have to take advantage of the hints available in the preceding words.

Predicting the next word is not easy /2

- There are easy predictions (“learnt by heart”):

  The former president of the United States, Donald $\rightarrow$

Predicting the next word is not easy /2

- There are easy predictions (“learnt by heart”):

  The former president of the United States, Donald $\rightarrow$ Trump

Predicting the next word is not easy /3

- There are predictions where a more complex processing ("reasoning") is needed:

  As the king only had a daughter, the next monarch, instead of a king, would be a $\rightarrow$

Predicting the next word is not easy /3

- There are predictions where a more complex processing ("reasoning") is needed:

  As the king only had a daughter, the next monarch, instead of a king, would be a $\rightarrow$ queen

(correct predictions of a relatively small language model, `EleutherAI/gpt-neo-1.3b`)

Predicting the next word is not easy /4

What happened? To try and predict the next word, the language model has to perform *abstract manipulations* with words without knowing their meaning.

- It tries to establish relationships between the words.
- We will see that it does so by creating *abstract representations* of the words.
- It then uses the relationships among these representations to try and predict the next word.

These manipulations may be perceived as being analogous to the ones performed by humans: this is why they are often called *(generative) artificial intelligence*.

Index

Artificial neurons /1

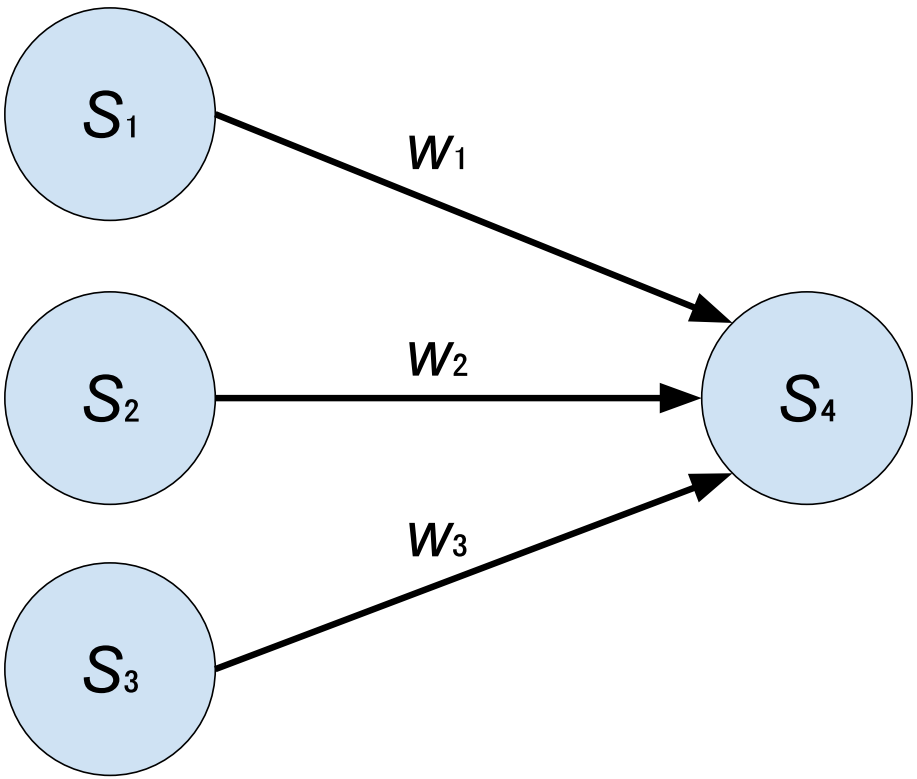
Why are large language models called neural?

- Because they are large networks of *artificial neurons*, *simulated* by computer (not physically built).
- The *activation* (excitation) of a neuron depends on the activation of neighbouring neurons and on the properties of mutual connections.
- The sign and the magnitude of the *weights* of these connections determine the behaviour of the network:
  - Neurons connected with a positive weight tend to excite or inhibit simultaneously.
  - The neurons connected with a negative weight tend to be in opposite activation states.
  - The effect of the interaction increases with the magnitude of the weight.

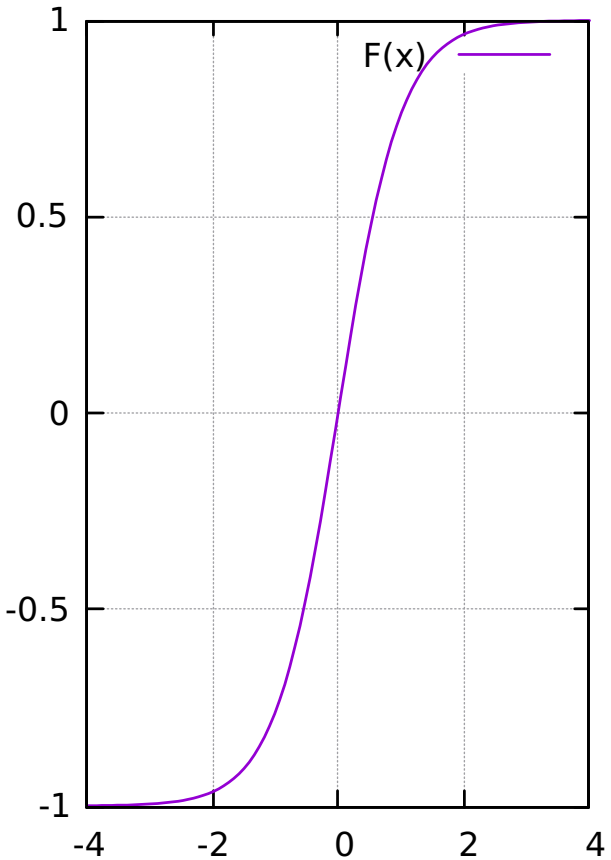
Artificial neurons /2

- The *training* of the artificial neural network sets the weights at the values necessary to ensure a specific behaviour:
  - specific excitation or inhibition patterns.

Artificial neurons /3



$$S_4 = F\,(w_1 \times S_1 + w_2 \times S_2 + w_3 \times S_3)$$



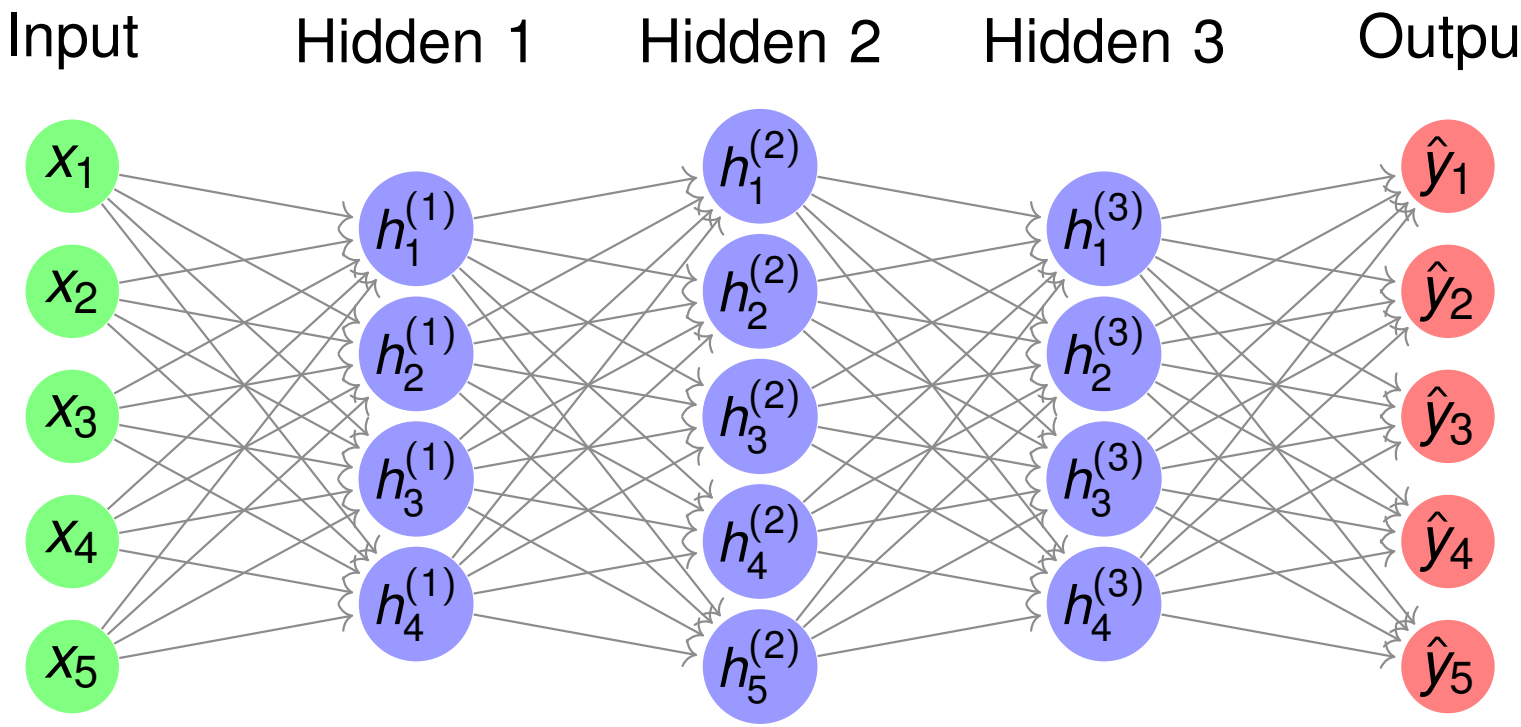
Artificial neurons /4

Many different *activation functions* $F()$ are possible:

- The one in the previous slide is a classical *hyperbolic tangent* function: $F(x) = \tanh(x)$, a sigmoid with values in $[-1, 1]$.
- There is also the logistic function $F(x) = \frac{1}{1+\exp(-x)}$, another sigmoid with values in $[0, 1]$.
- More modern functions are
  - the ReLU (*rectified linear unit*) function $F(x) = \max(0, x)$,
  - the "swish" function $F(x) = \frac{x}{1+\exp(-x)}$

Sometimes activation functions are used in connection with *linear gating* mechanisms involving a *layer* of neurons, and element-wise matrix multiplications.

Neural networks



A neural network with 5 inputs, 4 neurons in the 1st hidden layer, 5 neurons in the 2nd hidden layer, 4 units in the 3rd hidden layer and 5 output neurons.

Representations /1

When neural networks are used for *language tasks*:

- Each word in the vocabulary is assigned a number.
  - For example, number 34 is assigned to the word *house*.
- All inputs are turned off (signal=0.0) except the one corresponding to the word which is turned on (signal=1.0).
- The size of the input vocabulary is limited to a certain number of words: as many as inputs.
- Outputs are interpreted as probabilities.
  - The activation of output neuron number j is proportional to the probability of the j-th word in the vocabulary.

Representations /2

- When trained on a language task, the activation values of neurons in a layer form *representations* (nowadays called *embeddings*) of the information that is being processed.
- For example, the *vector*

$$(0.35, 0.28, -0.15, 0.76, \ldots, 0.88)$$

could be the representation of the word *study*, and the *vector*

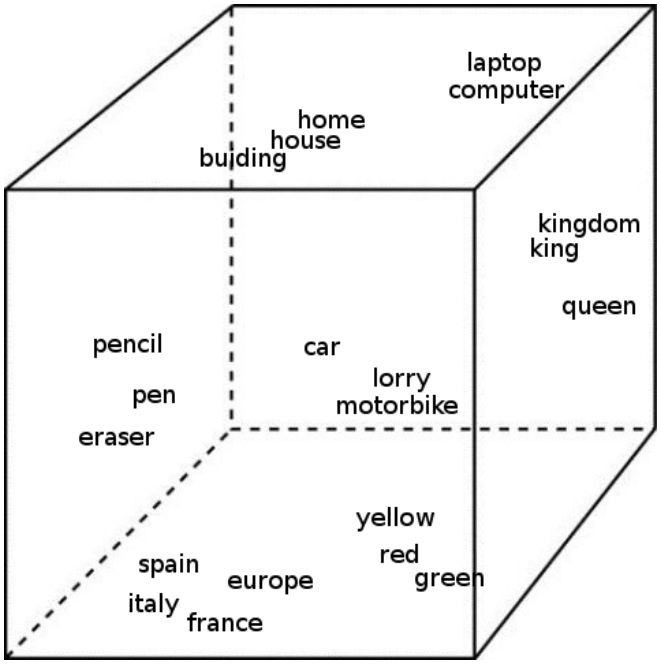
$$(0.93, -0.78, 0.22, 0.31, \ldots, -0.71),$$

that of word *cat*.

- We say that words are *represented* in a *vector space*.

Representations /3

Imagine lexical representations (“word embeddings”) using only three neurons (is not easy to imagine spaces with more than 3 dimensions):



Words with similar interpretations are close to each other.

Representations /4

- Learnt representations are usually such that one can perform semantic arithmetic (adding and substracting activation values neuron by neuron):[2]

$$[\text{king}] - [\text{man}] + [\text{woman}] \simeq [\text{queen}]$$

- This is an indication that the network does try to represent the relationships among words (the “semantics”).

[2] as one would do with columns of numbers in a spreadsheet

Neural language models/1

Neural language models

- are *predictors*: they predict words one by one.
  - Not too diferently from how your smartphone predicts the next word as you type a message.
- More precisely, *they estimate the probability* or the *likelihood* $p(t)$, *for each possible word* t of the vocabulary V, that this word has to appear in the current position.

Neural language models/2

In neural language models:

- The *prediction* process selects the most likely words.
- The likelihood of word t at position k, $p_k(t)$, depends on the sequence of the M words preceding it:

$$p_k(t \mid t_{k-M}\ t_{k-M+1}\ t_{k+M-2}\ \cdots\ t_{k-2}\ t_{k-1})$$

Not really words, but *subwords*.

Language models work with lexical units or *tokens* that can be words or "subwords":

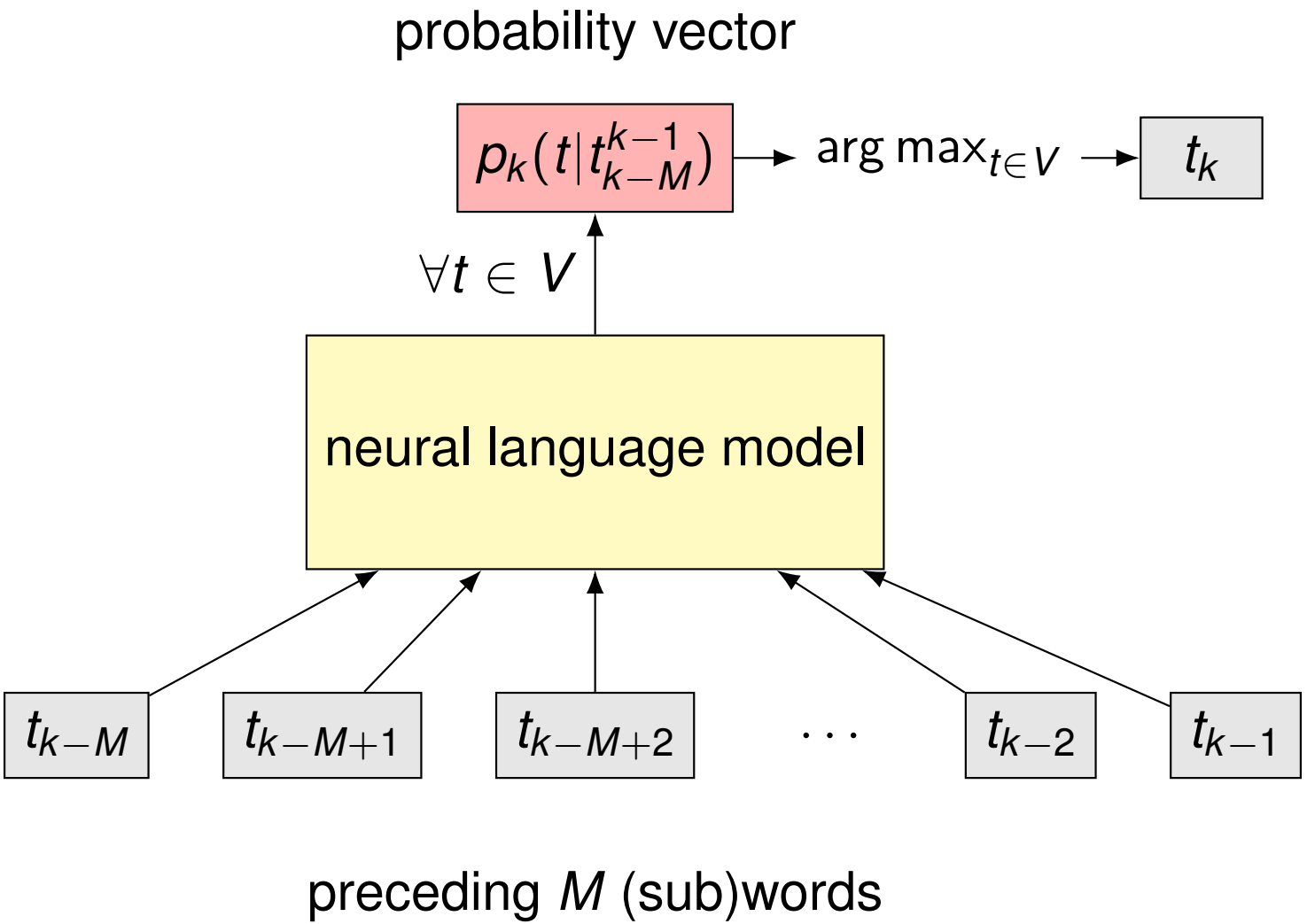
- For frequent words, the token is the whole word
- For less frequent words, the word is split in smaller (subword) units:

```
The acronym T- ES- C- RE- AL encompass- es
  a number of futur- ist philosophi- es .
```

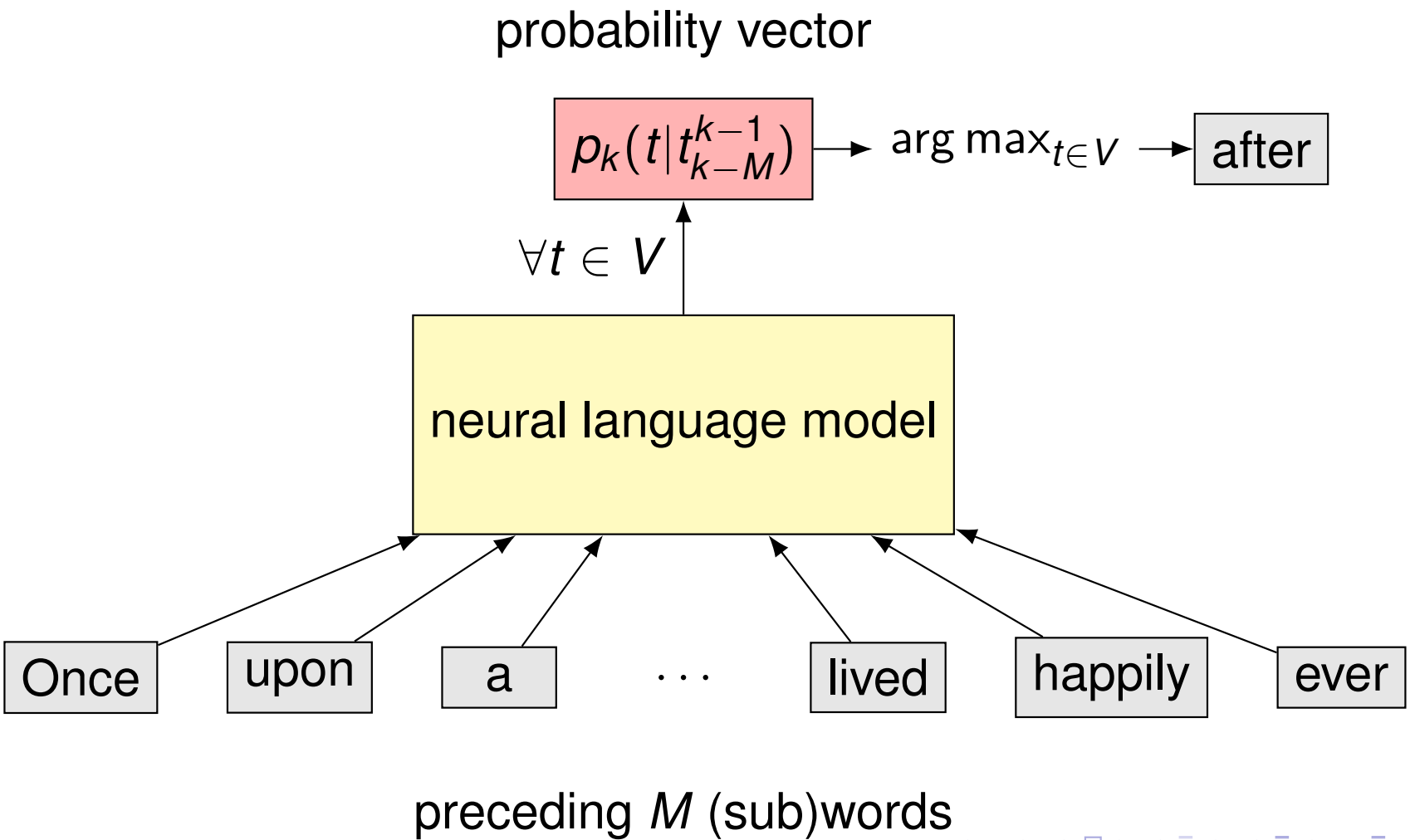
- This segmentation is automatically learned from monolingual text in an unsupervised way (*byte pair encoding*, *SentencePiece*, etc.).

This avoids huge networks, as the network has one input and one output for each (sub)word t in the vocabulary V.

The language model as a black, er. . . yellow box /1



The language model as a black, er. . . yellow box /2



The language model as a black, er. . . yellow box /3

In a language model trained on bedtime stories,

$$p_k(\text{“after”}|\text{“Once upon a. . . lived happily ever”})$$

should be higher than

$$p_k(\text{“before”}|\text{“Once upon a. . . lived happily ever”})$$

In fact, it should ideally be higher than that for any other (sub)word t in V:

$$p_k(\text{“after”}|\text{“Once upon a. . . lived happily ever”}) > p_k(t|\text{“Once upon a. . . lived happily ever”})$$

for any other t in the vocabulary.

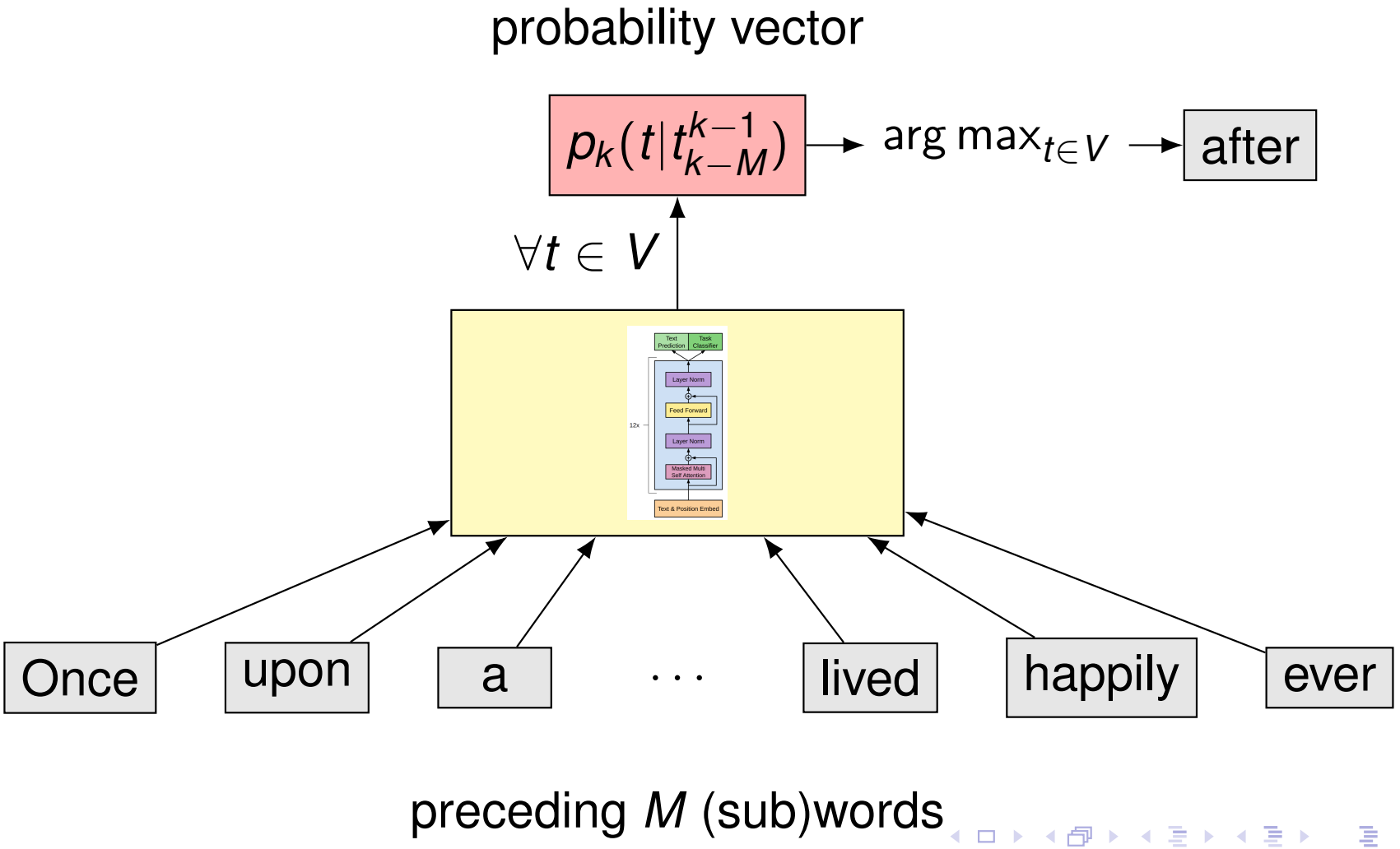
Index

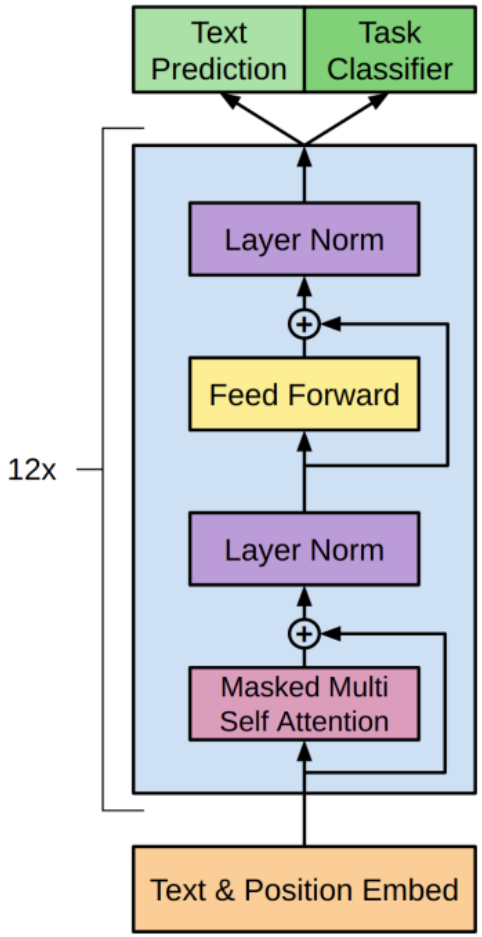
Transformers

The calculations made inside the yellow box may be performed using a variety of neural *architectures*, but the most common is now the *transformer*.[3]

[3] Vaswani, N., et al. (2017) “Attention is all you need”. In: *Advances in Neural Information Processing Systems*, pp. 6000–6010.

A *transformer* inside the box





Radford et al. (2018) “Improving Language Understanding by Generative Pre-Training” (informe intern d’OpenAI.)

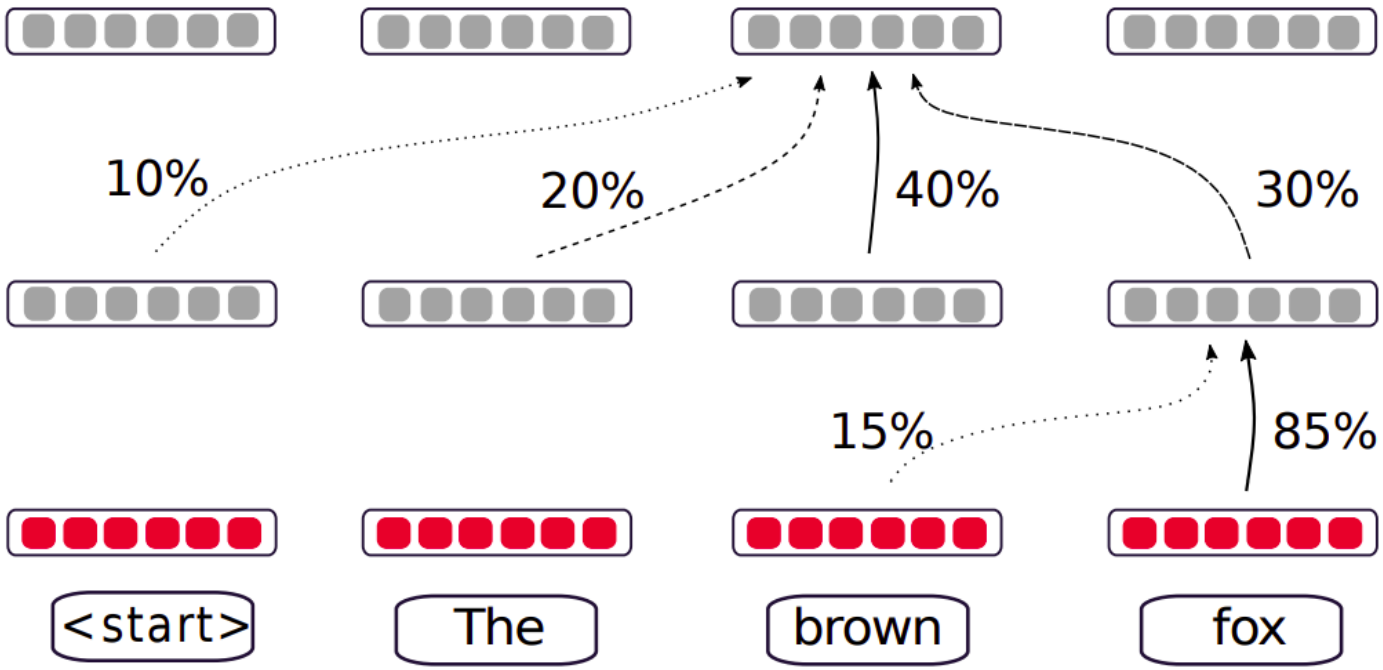
Transformers

A *transformer* works using *attention* mechanisms, and does so in stages (each stage uses a set of multi-layer neural networks):

- In each stage, *context* is incorporated to the representation of each token by *paying selective attention* to the representations of tokens in the preceding stage.
- In the last stage, attention is paid to all of the representations to generate the output:
  - Probabilities for each possible (sub)word in the next position (prediction)
  - Labels for other tasks (we'll see that in a little while).
- The number of stages is typically 12.
- The representations are vectors of, say, 2,048 components.[4]

[4] A round number in computing, as $2^{11} = 2{,}048$.

Transformers



[5] From Pérez-Ortiz, J.A., Forcada, M.L., Sánchez-Martínez, F., "How neural machine translation works", in Kenny, D., ed., *Machine translation for everyone: Empowering users in the age of artificial intelligence*, 141–164

Attention/1

A metaphor to make sense of the intricate maths:[6]

- Each stage is a generation in the history of a population.
- Each of the 2,048 representations in a given stage is an individual in the corresponding generation.
- The representations (individuals) in a stage (generation) use a kind of *online dating service* to later *mate* and generate the representations in the next stage (offspring, the next generation).

[6] Inspired in `https://youtu.be/aL-EmKuB078`, video of DotCSV.

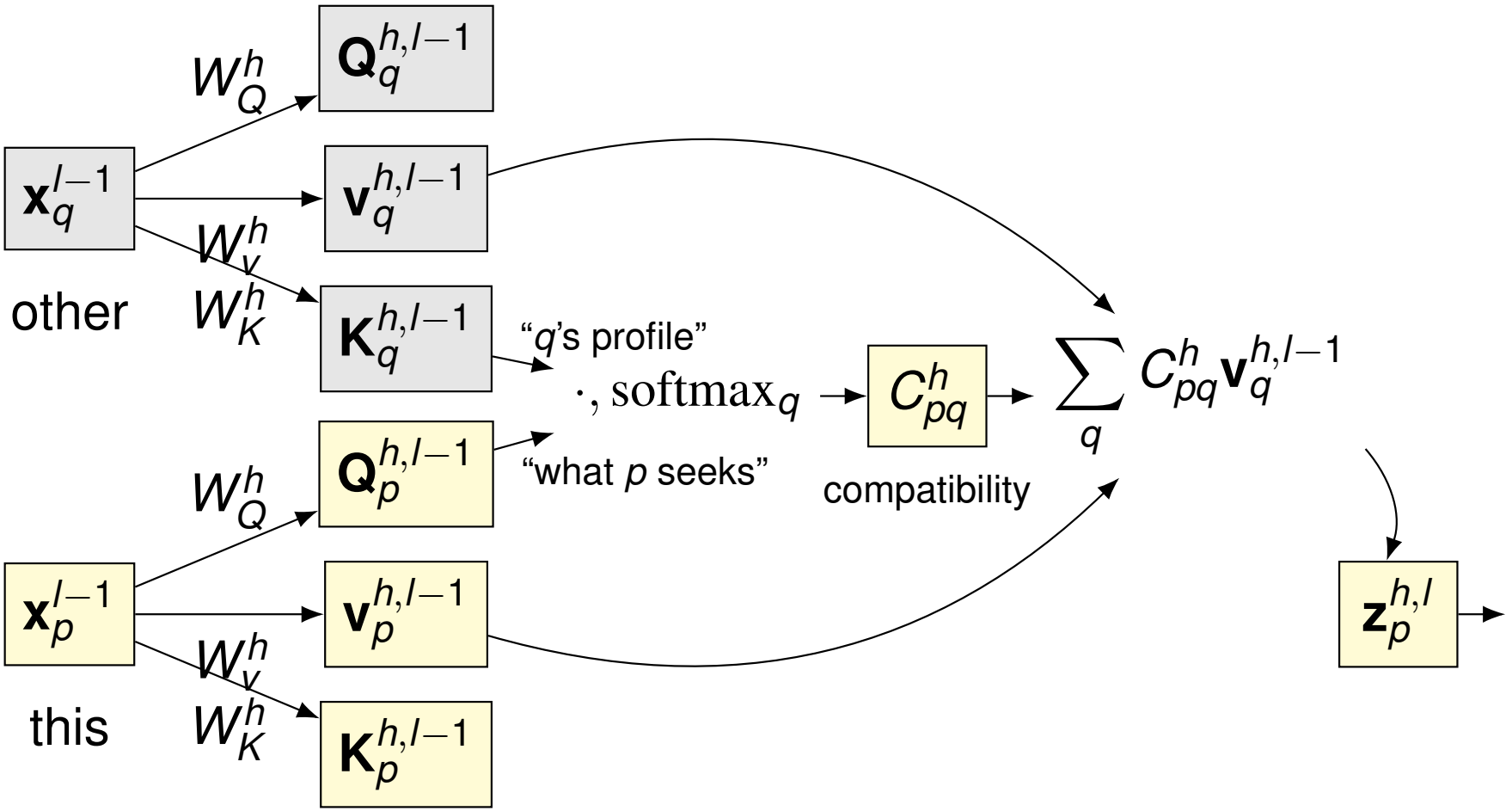
Attention /2

Matching phase:

- Each representation (individual) in each stage:
  - offers a profile of himself to the rest of representations (*key* vector);
  - generates a profile of the mates wanted (*query* vector);
  - generates genetic material that will be combined (*value* vector);

  (three vectors)
- From a representation's query and the profile of each of the other representations, a percentage of compatibility (attention) is generated.

Attention /2'


W_Q^h
$\mathbf{Q}_q^{h,l-1}$
$\mathbf{x}_q^{l-1}$
$\mathbf{v}_q^{h,l-1}$
W_V^h
other
W_K^h
$\mathbf{K}_q^{h,l-1}$
"q's profile"
$\cdot$, softmax$_q$
C_{pq}^h
$\sum_q C_{pq}^h \mathbf{v}_q^{h,l-1}$
$\mathbf{Q}_p^{h,l-1}$
"what p seeks"
compatibility
W_Q^h
$\mathbf{x}_p^{l-1}$
$\mathbf{v}_p^{h,l-1}$
W_V^h
this
W_K^h
$\mathbf{K}_p^{h,l-1}$
$\mathbf{z}_p^{h,l}$

Wait! Did you say *softmax*?

- *Softmax* is one possible way of normalizing a vector so that their components add to one, as probabilities should:

$$\text{softmax}_{1\leq i\leq n}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}$$

- It's not the only one.
- A whole family exists of normalizing functions ($^{\alpha}$*entmax*, Peter, Niculae & Martins 2019, where *softmax* = ^{1}*entmax*).

Attention/3

Breeding phase:

- The genetic material generated by the representation is passed on to the next stage (generation)
  - it is a mixture of its own genetic material and that of other representations;
  - the percentage in the mix is the compatibility percentage.
- Representations in the next stage (generation) are generated from this genetic material.

Attention/3'

- The contribution $\mathbf{z}_p^{h,l}$ of head h to the p-th representation in the next level $\mathbf{x}_p^l$ is computed by weighting the values $\mathbf{v}_q^{h,l-1}$ of other q using the compatibilities C_{pq}^h:

$$\mathbf{z}_p^{h,l} = \sum_q C_{pq}^h \mathbf{v}_q^{h,l-1}$$

- The $\mathbf{z}_p^{h,l}$ vectors for each head are concatenated into a larger vector $\mathbf{Z}_p^l = \mathbf{z}_p^{1,l} \oplus \mathbf{z}_p^{2,l} \oplus \ldots \mathbf{z}_p^{H,l}$.
- The vector is processed by a two-layer neural network to produce $\mathbf{x}_p^l$.

Attention/4

To generate

- the offered profiles, the queries and the genetic material from each representation (individual) in the current generation,
- the percentage of compatibility (attention) from offered profiles and queries, and
- the representations (individuals) in the next stage (generation) from the genetic material

one uses neural networks that have been trained so that the *transformer*

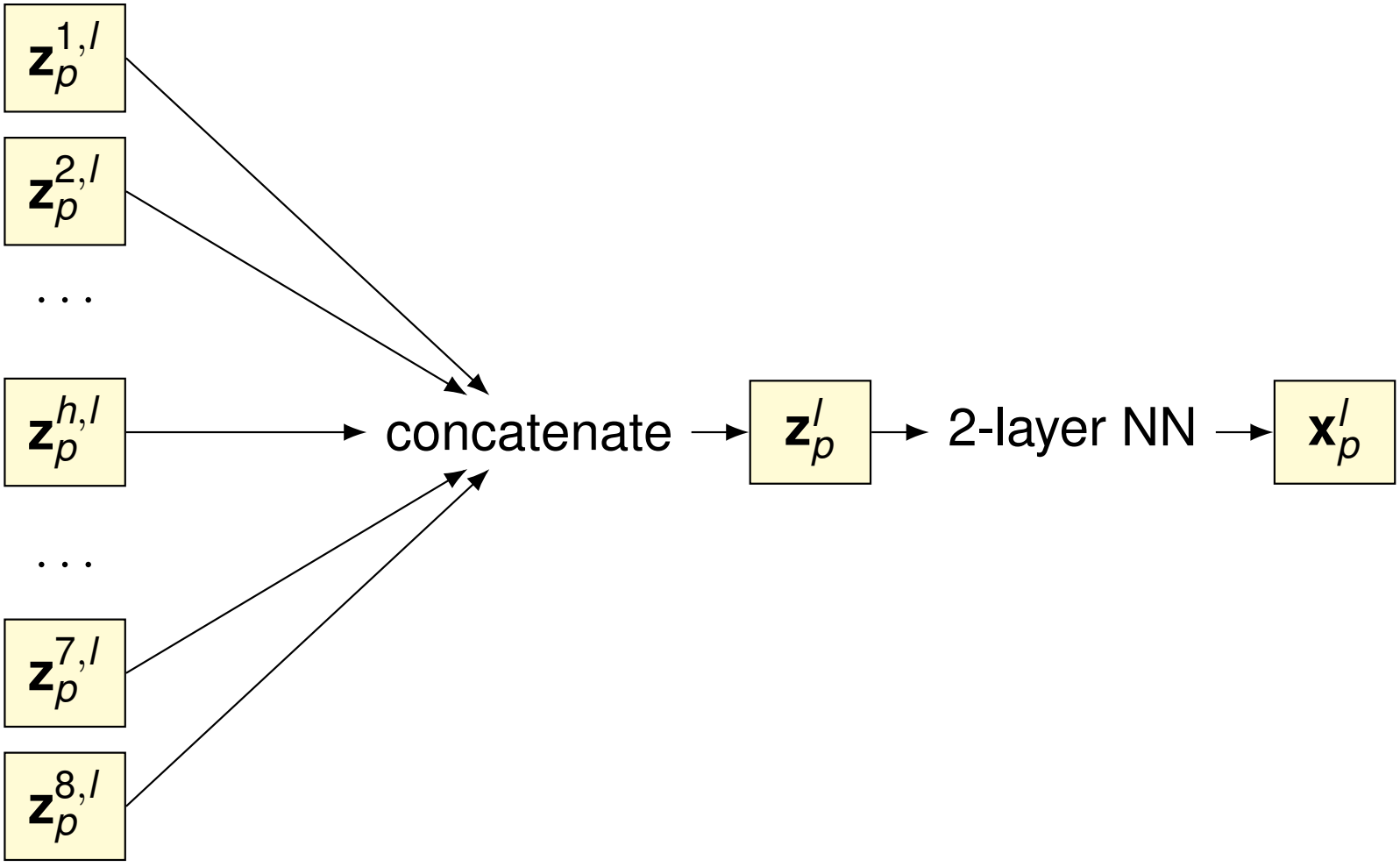
- produces the desired outputs (probabilities of each token in the next position)
- from the inputs (the preceding M tokens).

Attention/5

Everything is really a bit more complicated:

- The process described occurs typically in 8 rounds (*heads* in *transformer* parlance) in each stage.
- The initial representations of input tokens are *watermarked* with information about their position in the context window.
- There are additional direct connections (shortcuts) between layers.

Attention /5'


$\mathbf{z}_p^{1,l}$
$\mathbf{z}_p^{2,l}$
. . .
$\mathbf{z}_p^{h,l}$
. . .
$\mathbf{z}_p^{7,l}$
$\mathbf{z}_p^{8,l}$
concatenate
$\mathbf{z}_p^{l}$
2-layer NN
$\mathbf{x}_p^{l}$

Large language models: requirements

Training a large language model is not available to everyone:

- One has to obtain, manage, clean, and store a training corpus having thousands of millions of words of text.
- One needs high-performance computing facilities equipped with GPUs (*graphic processing units*).
  - The regular CPUs of laptop and desktop computers are too slow.
  - Neural training requires performing a lot of vector, matrix and tensor operations,
  - which are performed a lot more efficiently on GPUs.[7]
- The *models* can however be *optimized* so that they can be executed in ordinary computers.

[7] For example, one A100 GPU with 80 GB may cost $\simeq$ 20,000 EUR and uses 300 W of energy

Index

How are LLMs trained? /1

Training of LLMs occurs in three distinct phases:

- Phase #1: Text-only pretraining (language modelling proper).
- Phase #2: Supervised training in *semantic* labelling tasks.
- Phase #3: Supervised, then unsupervised *reinforcement learning* from *human feedback*.

How are LLMs trained /2

Training is setting the weights of all the connections so that the neural network executes a task.

Phase #1: *unsupervised pretraining*

- Task: predicting the next word ($\leftarrow$ language model)
- Goal: for each M-word block, maximize the probability assigned by the model to the word that actually comes next in the corpus.
- This probability has to be maximized for all the positions of in the training text corpus.

How are LLMs trained /3

- One may start with random weights or with those of a previously-trained model.
- The training algorithm computes a gradient $\frac{\partial P}{\partial w}$ of the probabilities P predicted for each word of the text with respect to each weight w connecting neurons.
  - That is to say, how much the probability varies with a small change in each weight w.
- After processing a certain number of examples, the weights update *proportionally* to their effect on the probability $\rightarrow$ *gradient ascent*

$$\text{new } w = w + \eta \frac{\partial P}{\partial w}$$

where η is the *learning rate*.

How are LLMs trained /4

The process repeats until

- a certain amount of time elapses, or
- a predetermined number of passes through the corpus have been completed, or
- the behaviour in a fraction of the corpus that has not been used to train (a *development* corpus) worsens.

Pause #1: pure text-only learning? /1

Let's pause and think.

- Some authors contend that this text-only training does not lead to semantic learning:
  - Cfr. Emily M. Bender's "Thought experiment in the National Library of Thailand"[8]
- But texts do encode knowledge (semantics) about real or hypothetical worlds: meaning $\rightarrow$ text.
- Which is later interpreted by humans using their context (incl. beliefs and expectations): text $\rightarrow$ meaning.

[8] https://t.ly/3uyLL

Pause #1: pure text-only learning? /2

- Up to recently, most text was generated by humans: we therefore attribute humanness to texts.
- This may therefore often seem intelligent, perceived as *making sense* and *computing meaning*:

prompt meaning (in context)	$\rightarrow$	prompt text	$\rightarrow$	LLM	$\rightarrow$	reply text	$\rightarrow$	reply meaning (in context)

Pause #1: pure text-only learning? /3

- (Pre-)Training an LLM to predict the next word forces it to represent text and compute in a way that complex predictions are made.
  - *Distributionally semantic* representations: meaning from company ("We served *kueh* to our guests"[9])
  - Predictions that *make sense* in the real or imaginary worlds described by texts.
- This could probably work even if words were *encrypted* and *decrypted*.
  - Provided that information in the internal structure of words is somehow kept.
- LLMs create intricate *semantic* devices to perform succesful text predictions.

[9] https://t.ly/DFlnQ

Pause #1: pure text-only learning? /4

Text-only manipulations are common in formal contexts:

- I have a rectangular field where trees are planted in 315 rows of 618 trees. How many trees I have?
  - Counts encoded as strings of digits: "`3 1 5`", "`6 1 8`"
  - Digits are placed one above the other
  - Digit pairs are looked up (multiplication tables, addition tables) and return one or two digits.
  - Blindly following preestablished rules, a table of characters is filled up.
  - The bottom row is decoded as a count.

			3	1	5
		×	6	1	8
		2	5	2	0
		3	1	5	
1	8	9	0		
1	9	4	6	7	0

Pause #1: pure text-only learning? /5

- Logic also works that way:

All wendot**s are** noracious
Justine **is a** wendot

$\Rightarrow$ Justine **is** noracious

- We can do this without knowing what a *wendot* is and what it means to be *noracious*.
- Logic is just mechanical text manipulation, like math.

How are LLMs trained /5

Phase #2: *supervised tasks*: assign a tag or class to a block of *M* input words.

- The training set is a corpus of *human-labeled* text blocks
- A special output layer is added to the transformer, which has an output neuron per class.
- Each task has its set of tags or classes.
- The mode is *retrained* ("*fine tuning*") so that it reproduces the labels in the training corpus.

How are LLMs trained /6

Tasks:[10]

Entailment: Premise + Hypothesis $\rightarrow$ T/F

Similarity: Text 1 + Text 2 $\rightarrow$ similarity score

Questions: Context + Question + Answer $\rightarrow$ correct?

Gramaticality: Text $\rightarrow$ gramatically correct?

One uses existing, publicly available corpora created during previous research for similar tasks.[11]

---

[10] Radford et al. (2018) "Improving language understanding by generative pretraining", OpenAI tech. report

[11] For example, GLUE (*General Language Understanding Evaluation*, `gluebenchmark.com`).

Pause #2: mixing in “semantic impurities”

Let’s pause again.
We are getting farther away from “text purity”

- Supervised retraining injects some human-labeled semantics into the transformer using an auxiliary task.
- Representations now change to try and succeed in these semantic tasks.

How are LLMs trained /7

Phase #3: *human feedback* (reinforcement learning with human feedback).[12]

- Using a corpus of queries and correct answers, the language model is supervisedly trained so that the answers are reproduced.
- Then, for each query (from another corpus) a sample of N random, different, likely responses are generated.
- Informants are asked to rank them by value (usefulness, acceptability, etc.).

[12] Ouyang et al. (2022) "Training language models to follow instructions with human feedback", OpenAI technical report

How are LLMs trained /8

- Another layer is added to the language model, such that, given a query and an answer, it generates a score (*reward model*, RM).
- The RM is trained so that the scores it generates reproduce the rankings given by informants as closely as possible.
- With a corpus of new queries, answers are generated, and the output of the RM is used to unsupervisedly retrain ("fine-tune") the LM.

Pause #3: “aligning” with goals /1

Yet another pause.
We are getting even farther away from “text purity”:

- The reward model learns to produce a score that mimic human rankings of acceptability, usefulness. . .
- Representations now change further to try and succeed in producing acceptable output.

Pause #3: “aligning” with goals /2

But please note: the resulting language models, used as *generative artificial intelligences*,

- learned some *semantics* when unsupervisedly trained to predict.
- were supervisedly trained on how to perform a few *semantic* tasks.
- were *reinforcement-trained* so that its replies were useful or acceptable in some way.

This is a rather indirect, partial way to train the LLM to meet the general *goals* of an *artificial intelligence*.

Bends ahead! 

Pause #3: “aligning” with goals /3

Compare these two uses of transformers:

Technology	**Nature of task**	**Reliability**	**Size**
Machine translation	Specific	High, controllable	Large
Generative Artificial Intelligence	General	Ample room for improvement	Huge

“Generative AI” is large, costly and less reliable than hype would like us to think.

Index

Some existing language models/1

There are a lot of language models out there:

- Some are *open and free*:
  - The training corpus is public.
  - The training strategy and settings are public.
  - You can download the model to use it on a local machine.
  - They may be used freely without any restriction.
- Other models are not:
  - You can use them for free or pay to use them.
  - You cannot download them to use them on a local machine.
  - You can download them but there are usage restrictions.
  - Not much is known about how they have been trained and on which corpus.

Some existing language models/2

Examples:

- Open and free:
  - The models by EleutherAI[13] (GPT-Neo, GPT-J, GPT-JT; the biggest has 20,000 million weights, trained on a public corpus called *The Pile* (886 GB of text).
  - Other models derived from GPT-2 such as GPT4All (virtual assistant)
  - The project RedPajama ($30 \cdot 10^{12}$ tokens of text) aims at reproducing LLaMA (view further down).

[13] `eleuther.ai`

Some existing language models/3

Examples:

- Not open / not free:
  - LLaMA (Meta, only allows non-commercial use) and LLaMA 2 (Meta and Microsoft) .
  - ChatGPT 3.5 (the famed ChatGPT: training partially unknown, free to use with constraints).
  - ChatGPT 4 (advanced paid version)
  - Google ~~Bard~~ Gemini

Some existing language models /4

To get a rough idea of the size of the corpora used:

- A not-too-long novel has about 100,000 words.
- A person can read 200–300 words per minute.
- One person can, therefore, read a novel in about 7 hours.
- *The Pile* [a rather small corpus] has 825 GB of text, about 50,000,000,000 words;
- That is to say, about 500,000 books:
- about 3,500,000 hours = about 400 years of continuous reading.

Index

Problematic effects of large language models /1

- They produce natural-looking, new text that seems written by humans.
  - They do not reproduce verbatim sections from the text corpus: they generate new texts.
- The responses are deceivingly *natural* but there is no guarantee of accuracy:
  - ChatGPT (as of November 24, 2023): “Does the Basque language make a T/V distinction?” $\rightarrow$ “No, the Basque language does not have a T/V (tu/vous) distinction. In Basque, there is a single second-person singular pronoun, which is "zu." [. . .]”. (wrong! There is *hi* (T) and *zu* (V).)

Problematic effects of large language models /2

- It is hard to know, with computation distributed across thousands of neurons and millions of connections, how they make the decisions needed to generate a response.
  - When asked to explain the reasoning behind one of their responses, these models produce an apparently introspective explanation, but it is just more generated text, disconnected from their actual inner workings.
    - Although probably partially grounded on recent context in the conversation.

Problematic effects of large language models /3

- Training them (or retraining them) requires:
  - managing and storing large quantities of text (thousands of millions of words);
  - specialized, expensive high-performance computer resources.

  Only a few companies or institutions can afford them $\rightarrow$ centralization, gatekeeping, dependency.
- Even just *using* them requires powerful computers endowed with sufficient memory.

Problematic effects of large language models /4

- Training and using these models uses a lot of energy and generates greenhouse gases, and building the computing facilities needed generates toxic waste $\rightarrow$ environmental effects are even harder on vulnerable societies.
- In spite of retraining and human feedback, they can still reproduce unjust biases encoded in the training sets.
- Smaller languages and vulnerable groups are usually underrepresented (or not represented at all):
  - ChatGPT (May 24, 2023 version) replies in Spanish or Catalan to questions in Occitan, even after being told about it.

Problematic effects of large language models /5

- Large language models will change many academic and professional sectors, as they simplify (industrialize?) the generation of text.
  - They can reformulate, expand with a specific intention, or summarize.
  - They can create drafts of texts with a literary or artistic intent (for example, a stanza of four verses on a subject).
  - They can draft computer programs in many programming languages.
  - They can draft multiple-choice questions from a syllabus text.
  - They may assist instructors during grading, after having given models and guidelines.

Index

Concluding remarks /1

- Neural language models generate text that:
  - is a continuation of previous text, or
  - is an answer to a question or *prompt*.
- They are *neural*: they are computer simulations of the behaviour of large networks of artificial neurons inspired in those of animals.
- A large language model:
  - is *pretrained* with large quantities of text and in an unsupervised way (predicting the next word);
  - it is *fine-tuned* with examples of textual tasks (supervised training, annotated with labels);
  - it is further *fine-tuned* with human feedback.

  As a result, their responses show an apparently intelligent behaviour.

Concluding remarks /2

- Large language models such as ChatGPT:
  - cannot be created by everyone (massive resources are needed) or executed by everyone (powerful resources are needed);
  - are not transparent and can be unjust;
  - have important environmental issues;
  - will radically change [are already radically changing] the work of professional and academic sectors involved in the creation and processing of texts.

These slides are free

Thank you very much!

Scan this QR code to obtain a copy of the slides

or go to **`https://t.ly/tvuLu`**.