6 – Limits and future of LLMs

IASD / MASH – LLMs course

Florian Le Bronnec

November 20, 2023

Table of Contents

Table of Contents

Standard tranformer model

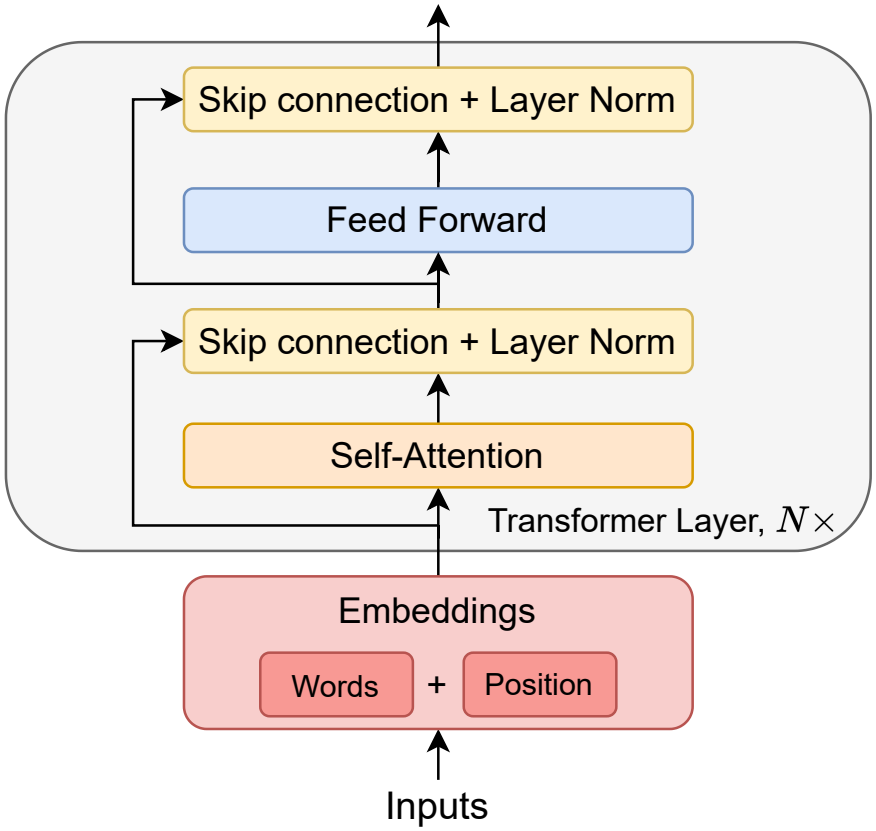


Figure 1: Standard stack of transformer layers.

BERT's MLM

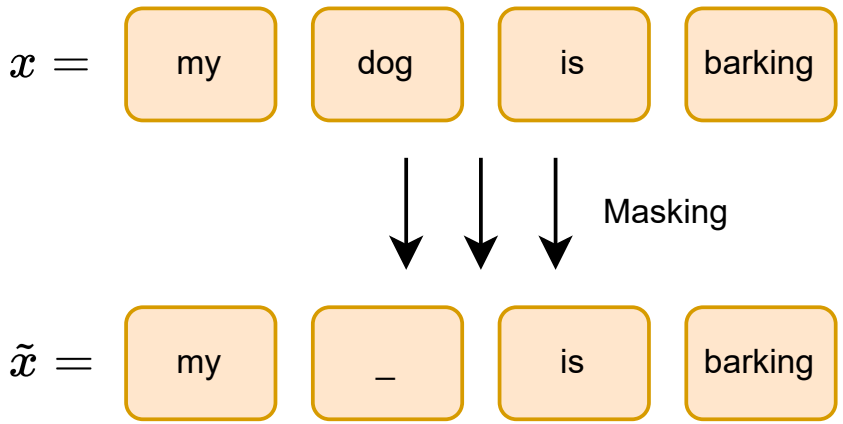


Figure 2: BERT masked language modeling.

$$\mathcal{L} = \sum_{\substack{w \in x \\ w \text{ is masked}}} -\log \mathrm{P}_\theta(w \mid \tilde{x})$$

BERT's Next sentence prediction

- Extract a sentence s_1 from a document $x \in \mathcal{D}$.
- With 50% chance, take s_2 the sentence following s_1.
- With 50% chance, take s_2 a random sequence from $\mathcal{D}$.

Predict if s_2 follows s_1:

$$\mathcal{L} = -\mathbf{1}_{s_2 \text{ follows } s_1} \log \mathrm{P}_\theta(s_1, s_2) - \mathbf{1}_{\text{random } s_2} \log(1 - \mathrm{P}_\theta(s_1, s_2))$$

Different kinds of attention

Attention in BERT

$$\begin{cases} s_{ij} &= \boldsymbol{q}_i^T \boldsymbol{k}_j \in \mathbb{R},\ 1 \leq j \leq \boldsymbol{L}, \\ \boldsymbol{\alpha}_i &= \text{Softmax}(\boldsymbol{s}_i) \in \mathbb{R}^{\boldsymbol{L}}, \\ \boldsymbol{y}_i &= \sum_{j=1}^{\boldsymbol{L}} \alpha_{ij} \boldsymbol{v}_j \in \mathbb{R}^d. \end{cases}$$

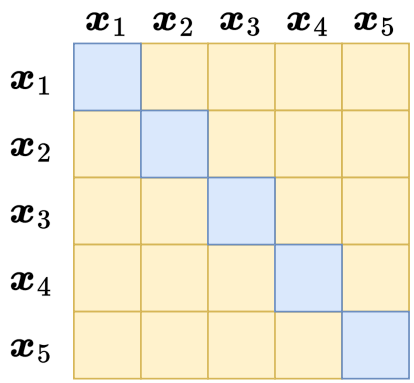


Figure 3: **Bidirectional** attention, tokens attend to every token.

Attention for generation

$$\begin{cases} s_{ij} &= \boldsymbol{q}_i^T \boldsymbol{k}_j \in \mathbb{R},\ 1 \leq j \leq \boldsymbol{i}, \\ \boldsymbol{\alpha}_i &= \text{Softmax}(\boldsymbol{s}_i) \in \mathbb{R}^{\boldsymbol{i}}, \\ \boldsymbol{y}_i &= \sum_{j=1}^{\boldsymbol{i}} \alpha_{ij} \boldsymbol{v}_j \in \mathbb{R}^d. \end{cases}$$

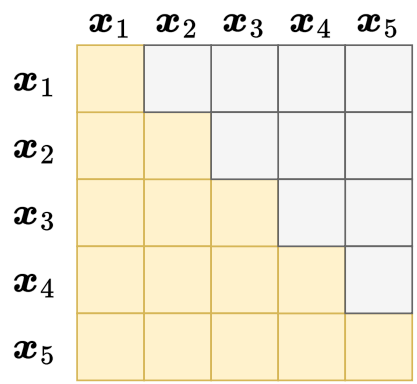


Figure 4: **Unidirectional** attention, tokens can only attend backward.

Generative models pre-training

$$\log \mathrm{P}_\theta(x) = \sum_{i=1}^{L} \log \mathrm{P}_\theta(x_i \mid x_{<i}).$$

Next token prediction objective.

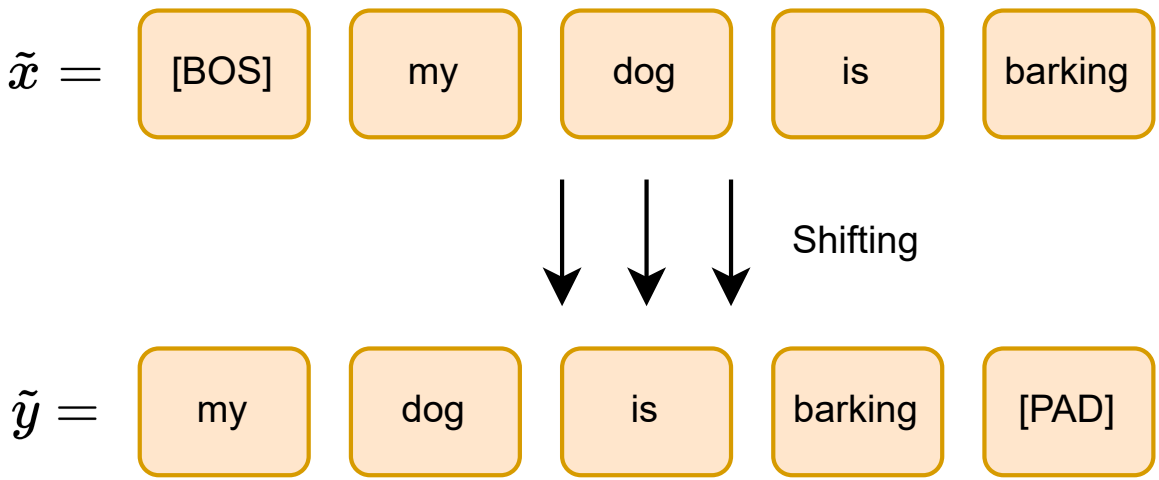


Figure 5: Generative models pre-training.

Encoder-decoder with cross-attention

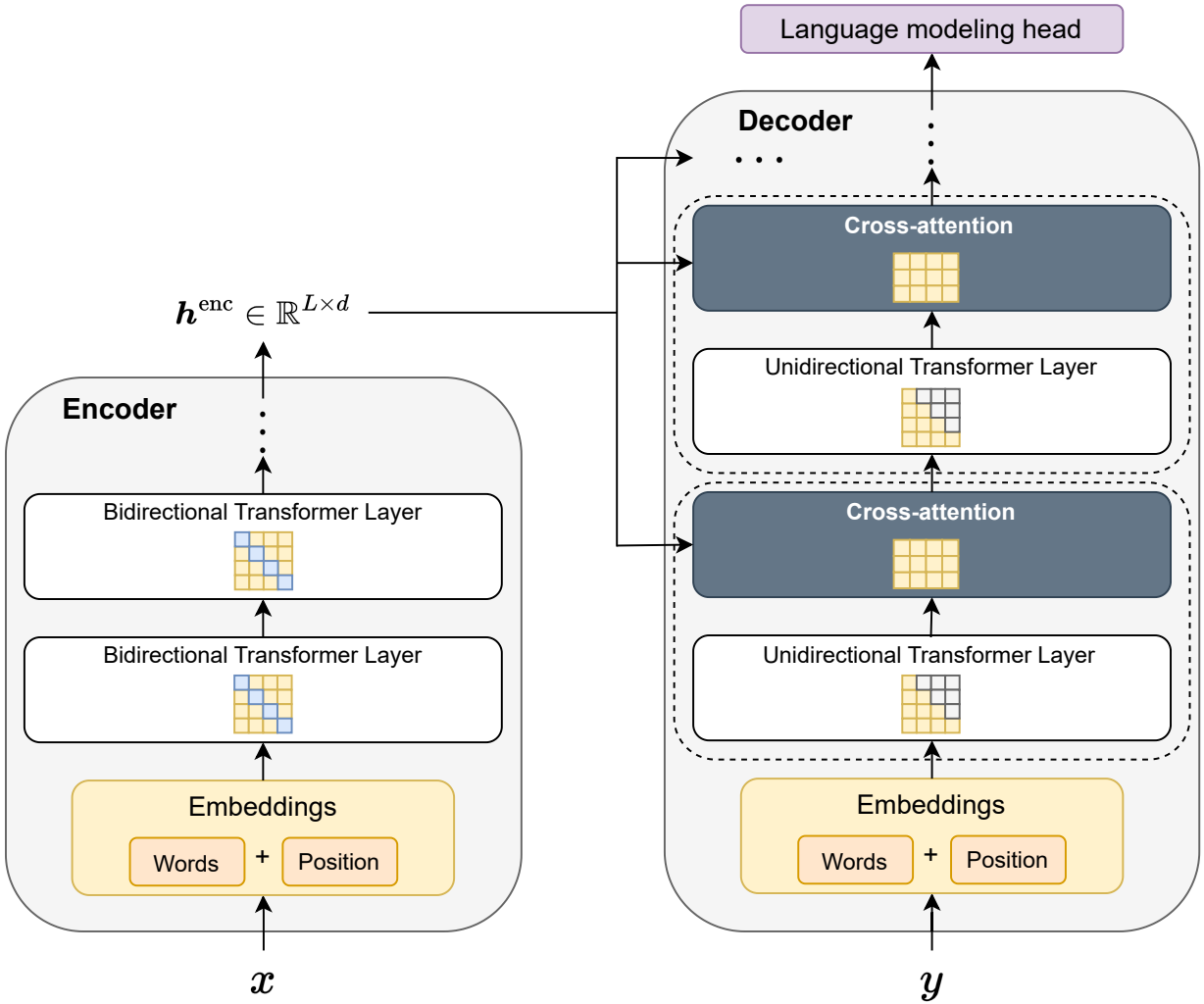


Figure 6: Encoder-decoder model with cross-attention layers.

BART's pre-training

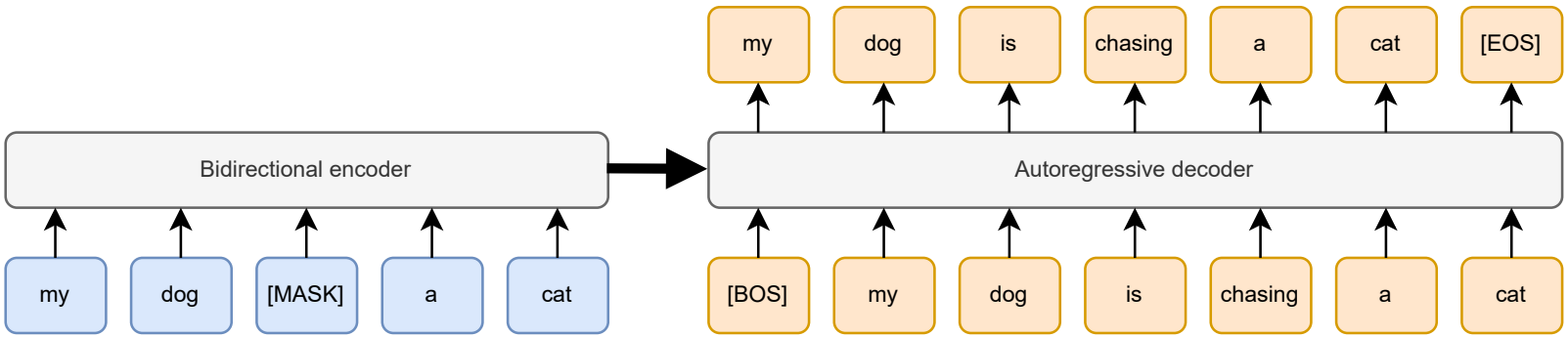


Figure 7: Denoising with BART.

Finetuning

Finetuning to specific supervised tasks.

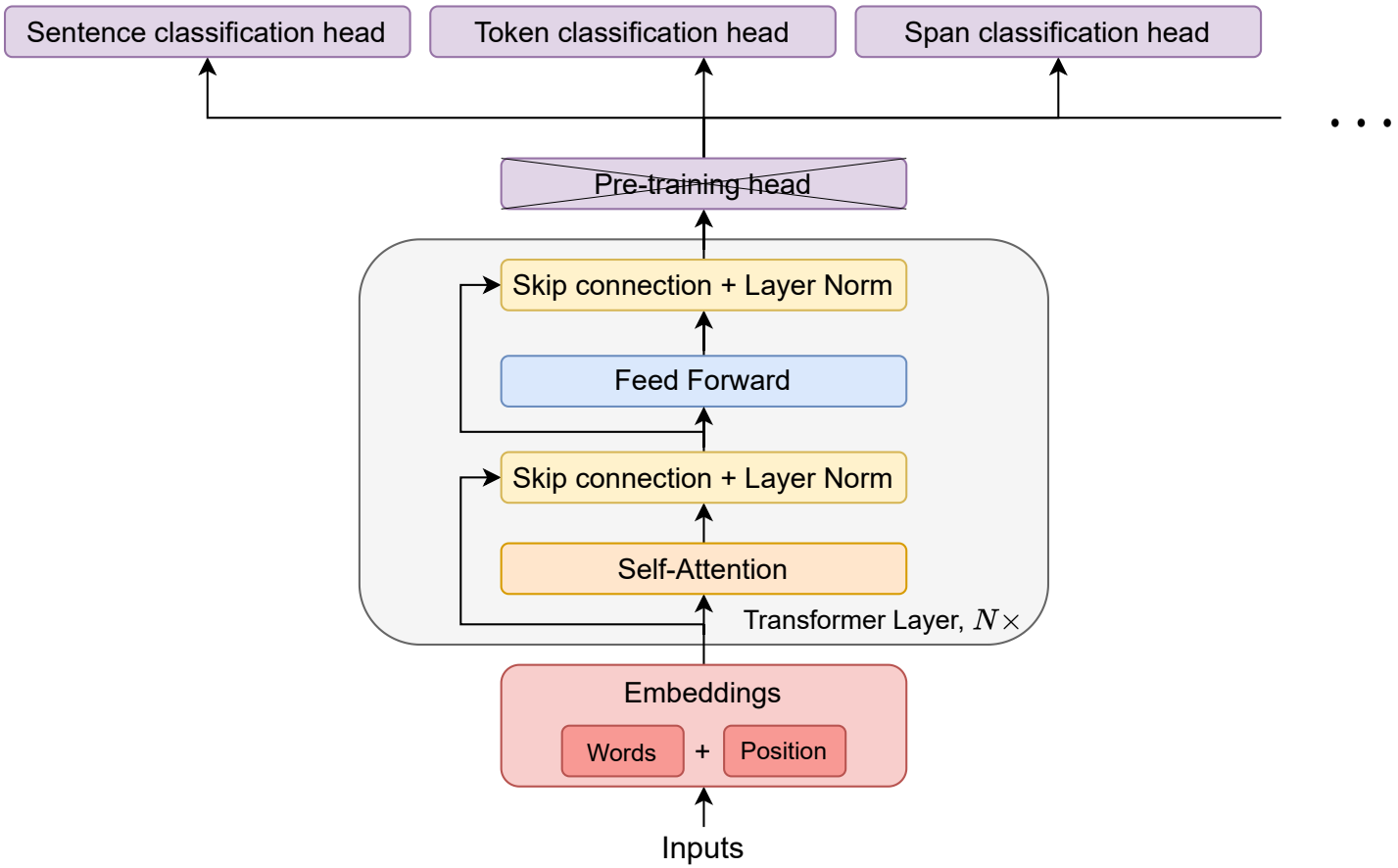


Figure 8: Switching from pretraining to finetuning.

Table of Contents

What's under the hood?

Impressive pretrained models

- BERT-like models achieve **better performances than humans** on the SuperGLUE dataset [14].
- BART-like models can **summarize texts, answer questions**, etc. [5].
- GPT3-3.5-4 are few-shot learners, they can answer questions based on their **general knowledge** [7].
- ChatGPT.

What's under the hood?

Impressive pretrained models

- BERT-like models achieve **better performances than humans** on the SuperGLUE dataset [14].
- BART-like models can **summarize texts, answer questions**, etc. [5].
- GPT3-3.5-4 are few-shot learners, they can answer questions based on their **general knowledge** [7].
- ChatGPT.

How?

- Data.
- Scaling the models.

What's under the hood?

Impressive pretrained models

- BERT-like models achieve **better performances than humans** on the SuperGLUE dataset [14].
- BART-like models can **summarize texts, answer questions**, etc. [5].
- GPT3-3.5-4 are few-shot learners, they can answer questions based on their **general knowledge** [7].
- ChatGPT.

How?

- Data.
- Scaling the models.

$\Longrightarrow$ What does it represent in practice?

Data

Model	Params	Context	Batch	Steps
BERT [4]	355M	512	256[1]	1M
BART [5]	406M	1024	8000	500K
LLama2 [20]	7-13-70B	4096	4000[1]	500K

Table 1: Comparison of NLP models. Steps are the number of training steps.

[1] Batch size were indicated in terms of number of tokens, I approximately converted it to number of documents.

Data

Model	Params	Context	Batch	Steps
BERT [4]	355M	512	256[1]	1M
BART [5]	406M	1024	8000	500K
LLama2 [20]	7-13-70B	4096	4000[1]	500K

Table 1: Comparison of NLP models. Steps are the number of training steps.

Demo!

[1] Batch size were indicated in terms of number of tokens, I approximately converted it to number of documents.

Pretraining in practice

Biggest NVIDIA GPUs are $\sim$ **8**$\times$ bigger than Colab's GPUs.

Even with bigger GPUs, processing batches of **8000 documents** is an issue. How can we do that in practice?

Pretraining in practice

Biggest NVIDIA GPUs are $\sim \mathbf{8}\times$ bigger than Colab's GPUs.

Even with bigger GPUs, processing batches of **8000 documents** is an issue. How can we do that in practice?

$\Longrightarrow$ Let's review some methods to perform these heavy trainings.

Data parallelism

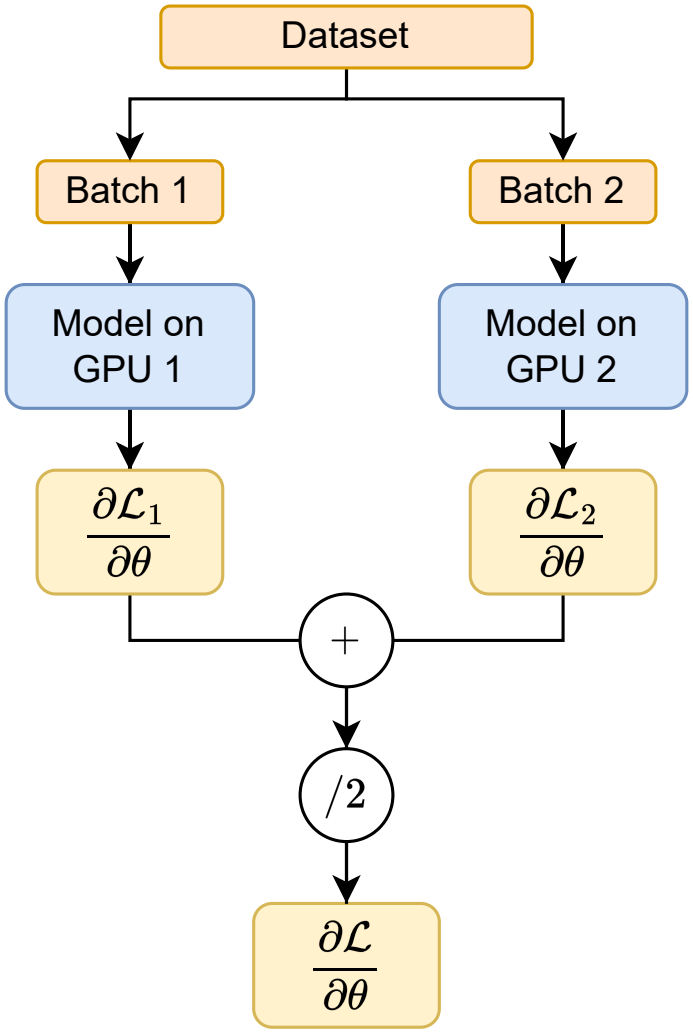


Figure 9: Data parallelism.

Gradient accumulation

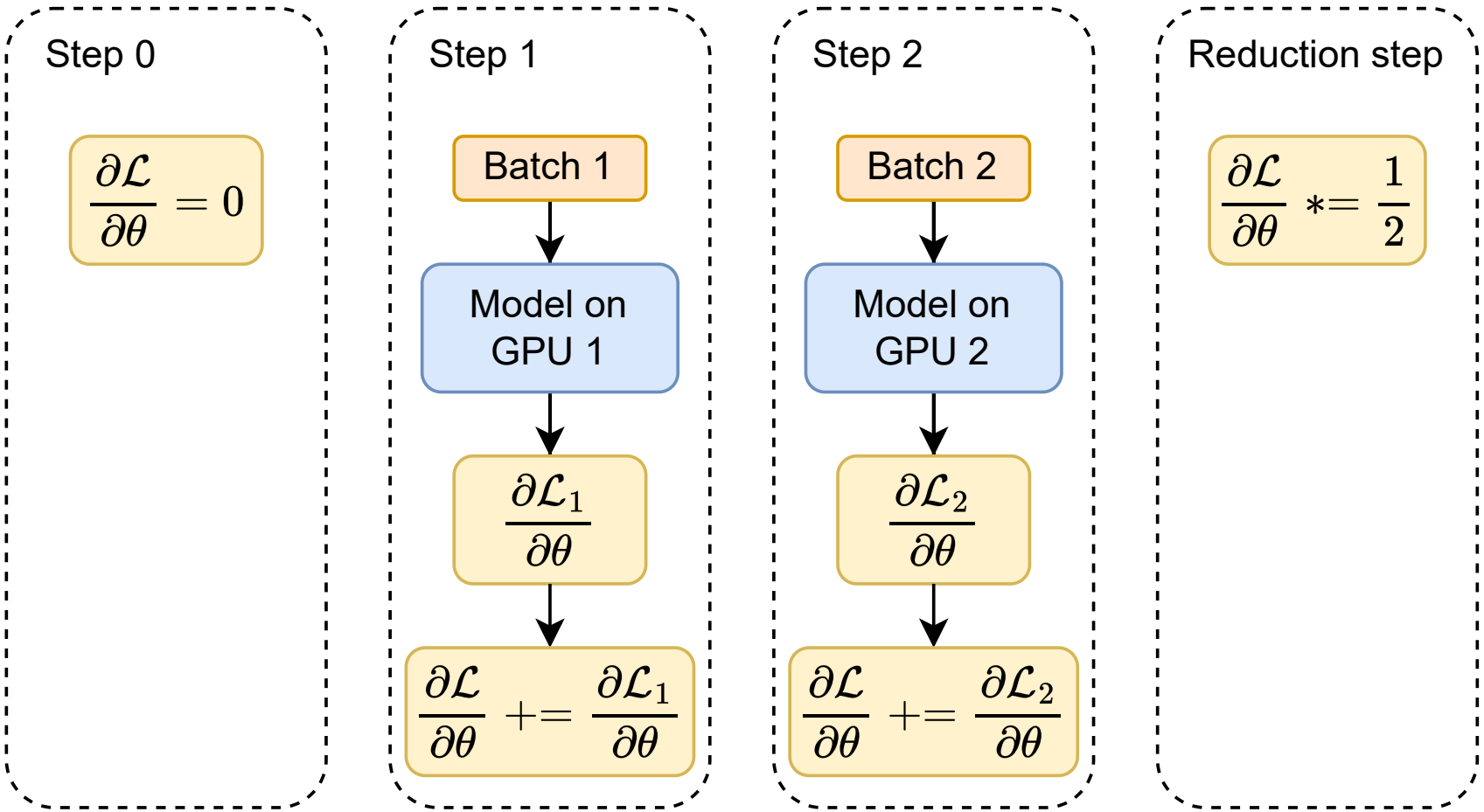


Figure 10: Gradient accumulation.

Summary

Data parallelism and **gradient accumulation** are two very common methods in NLP (and deep learning) to increase the effective size of the batch.

Data Parallelism

Per Device Batch Size	Number of Devices	Effective Batch Size
32	1	32
32	2	64
32	4	128

Gradient accumulation

Batch Size	Accumulation Steps	Effective Batch Size
32	1	32
32	2	64
32	4	128

Comparison

Aspect	**Data Parallelism**	**Gradient Accumulation**
Parallelism	Yes, across GPUs	No, sequential computations
Several GPUs	Yes	Can work on 1 GPU
Time	$1 \times (\texttt{forward} + \texttt{backward})$ + communication across GPUs	$N \times (\texttt{forward} + \texttt{backward})$

Table 2: Comparison of Data Parallelism and Gradient Accumulation

Comparison

Aspect	**Data Parallelism**	**Gradient Accumulation**
Parallelism	Yes, across GPUs	No, sequential computations
Several GPUs	Yes	Can work on 1 GPU
Time	$1 \times (\texttt{forward} + \texttt{backward})$ + communication across GPUs	$N \times (\texttt{forward} + \texttt{backward})$

Table 2: Comparison of Data Parallelism and Gradient Accumulation

$\Longrightarrow$ Data parallelism is useful when you have **several GPUs** and want to **speed up** the training.

$\Longrightarrow$ Gradient accumulation is useful when you have **limited resources.**

Both can of course be combined (and are combined in practice).

In practice?

Everything is quite easy with PyTorch.

Demo!

Scaling the number of parameters

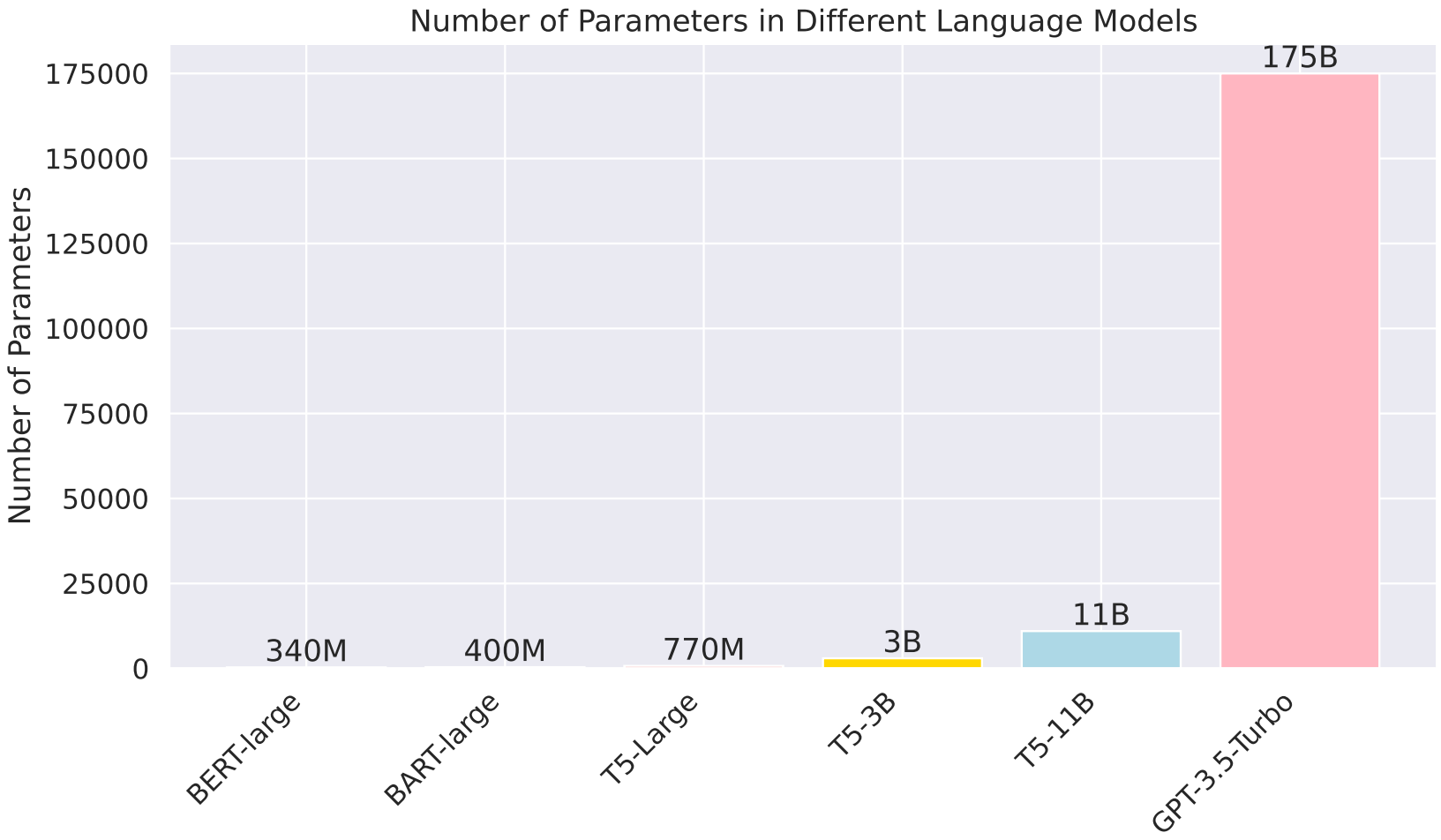


Figure 11: Different models and their scales.

What does it imply?

Demo on Bert size.

What does it imply?

Demo on Bert size.

BART on a GPU is $\sim$ 1.4GiB. GPT-3 is $\sim \times 500$ bigger than BERT.

Biggest GPUs are 80GiB.

$\Longrightarrow$ How does it fit?

What does it imply?

Demo on Bert size.

BART on a GPU is $\sim$ 1.4GiB. GPT-3 is $\sim \times 500$ bigger than BERT.

Biggest GPUs are 80GiB.

$\Longrightarrow$ How does it fit?

We're going to speak about **training** and **finetuning / inference**.

Model parallelism

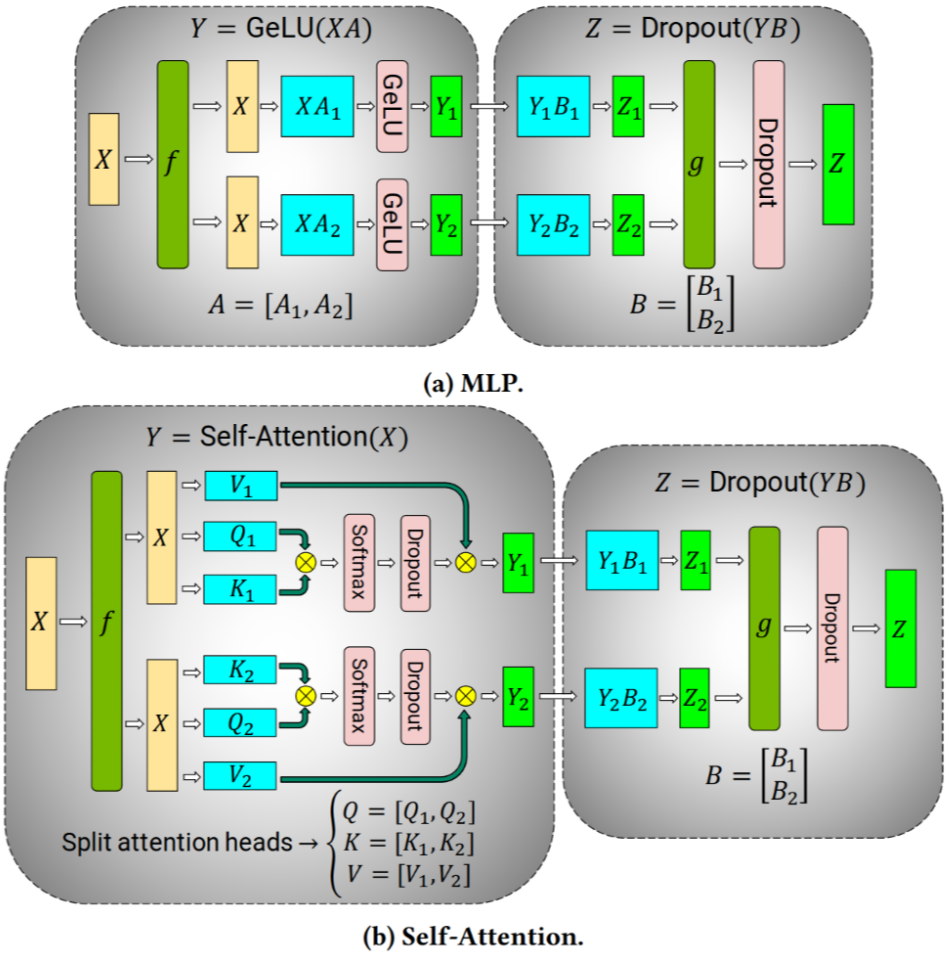


Figure 12: Idea: split the computations across several GPUs. Tensor parallelism, figure from [11].

Pretraining

For pretraining, there are some heavy hardware optimizations, like the ones presented in [11].

The idea is to split independant and costly operations across devices, then aggregate the final result.

Even if there are solutions, as a rule of thumb remind that **model parallelism is not easy**.

And for end-users?

Most people are not (and should not) be interested in what we presented in previous slides.

But still, we might want to use these models for at least:

- running inference as is,
- finetuning on a specific task.

$\Longrightarrow$ Let's describe some solutions for practitioners.

Some solutions: Quantization

GPUs standard precision is `float32`. A solution is to **reduce this precision**.

Data Type	Bit Width	Hardware Capability	Use for training
`float32`	32	General-purpose CPUs/GPUs	Yes
`float16`	16	GPUs with FP16 support	Yes
`bfloat16`	16	NVIDIA Ampere GPUs, TPUs	Yes
`int8`	8	CPUs, GPUs	No

Table 3: Non-exhaustive list of mixed-precision data types and hardware support.

Some solutions: Quantization

GPUs standard precision is `float32`. A solution is to **reduce this precision**.

Data Type	Bit Width	Hardware Capability	Use for training
`float32`	32	General-purpose CPUs/GPUs	Yes
`float16`	16	GPUs with FP16 support	Yes
`bfloat16`	16	NVIDIA Ampere GPUs, TPUs	Yes
`int8`	8	CPUs, GPUs	No

Table 3: Non-exhaustive list of mixed-precision data types and hardware support.

$\Longrightarrow$ In practice the models maintain their performances (empirical statement).

Be careful when using quantization

- Make sure the operations you use are well supported.
- Read the documentation, especially for training.
- There can be some instabilities (ex: T5 does not work with `float16`).

Reduce the cost of autodiff

Component	Memory Cost	Inference
Model Parameters	$O(P)$	Yes
Activations	$O(B)$	Yes / No
Gradients	$O(P)$	No
Optimizer States	$O(P)$	No

Table 4: Memory cost during training and inference.

Huge dependency on P, the number of parameters to update.

Reduce the cost of autodiff

Component	Memory Cost	Inference
Model Parameters	$O(P)$	Yes
Activations	$O(B)$	Yes / No
Gradients	$O(P)$	No
Optimizer States	$O(P)$	No

Table 4: Memory cost during training and inference.

Huge dependency on P, the number of parameters to update.

$\Longrightarrow$ Reduce the number of parameters to update!

Presentation on Adapters + LORA.

Break!

Table of Contents

Which length of texts can we process?

How many tokens can fit in the models?

There are 3 types of limitations:

- **architectural**, because of position embeddings of fixed size,
- **training setup**, the maximal length seen during training, independently of the position embeddings,
- **computational cost**, because the costs scale with the context length.

Attention is $\mathcal{O}(L^2)$

Let L be the input length.

$$\boldsymbol{Q}\boldsymbol{K}^T \in \mathbb{R}^{L \times L}.$$

Demo!

Attention is $\mathcal{O}(L^2)$

Let L be the input length.

$$\boldsymbol{Q}\boldsymbol{K}^T \in \mathbb{R}^{L \times L}.$$

Demo!

$\Longrightarrow$ **Quadratic** cost w.r.t L.

How can we avoid that?

Scaling the context

Simple idea: **reduce the size of the attention matrix**.

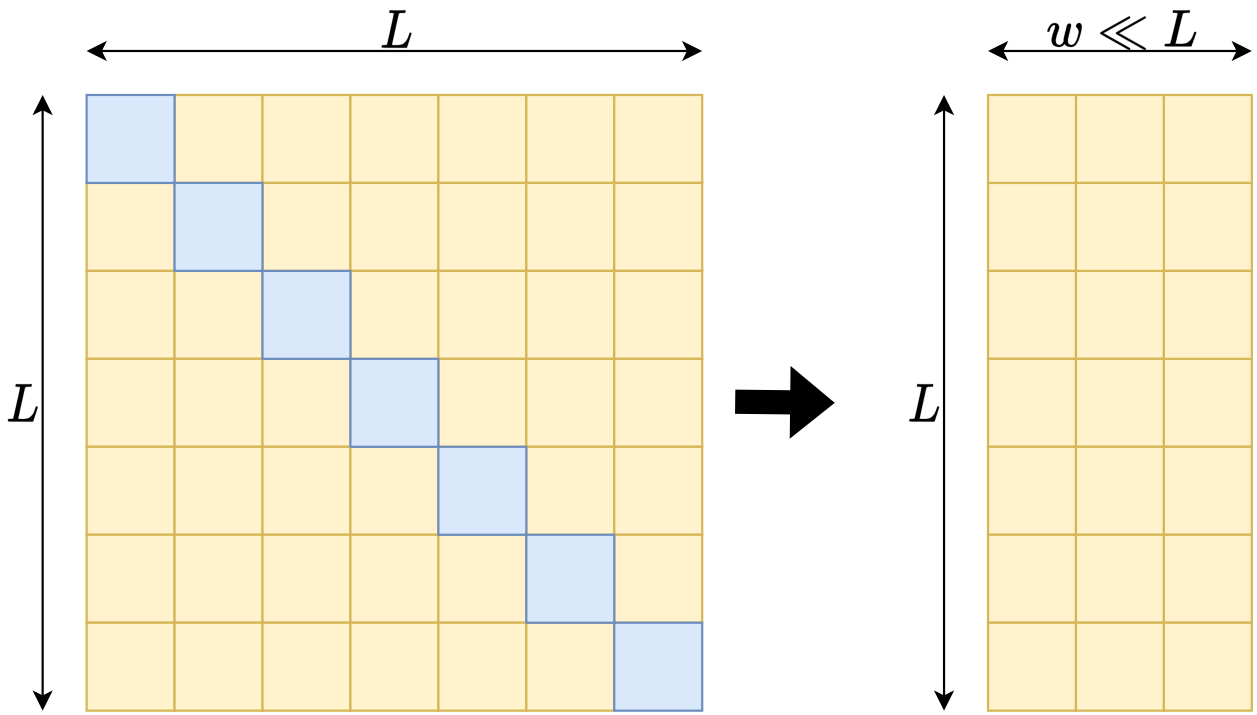
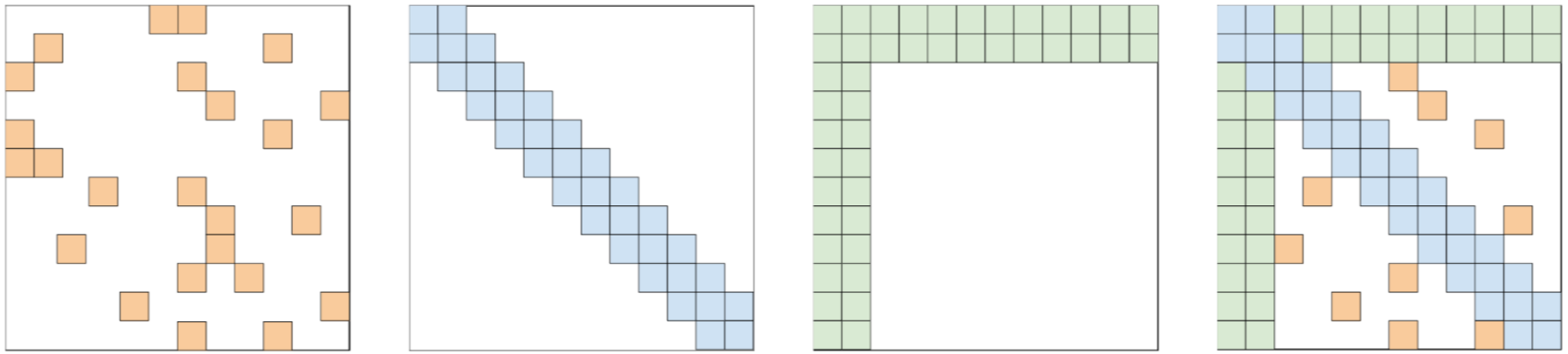


Figure 13: Reducing the size of the attention matrix.

We can therefore reach a $\mathcal{O}(Lw)$ memory cost.

Sparse attention



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 14: Sparse attention patterns, figure from [15].

Whole litterature on sparse transformers:

- LongT5 [17],
- BigBird [15],
- Longformer [6] (extends pretrained models),
- etc.

Other solutions

Hardware improvements

- FlashAttention [16] proposed recently a new CUDA kernel optimized for computing attenion on recent NVIDIA GPUs.

Other solutions

Hardware improvements

- FlashAttention [16] proposed recently a new CUDA kernel optimized for computing attenion on recent NVIDIA GPUs.

Stick to the model's base length

- 4k tokens is already a lot.
- Models have **not been pretrained on such lengths**.
- LLMs **do not use their full context** [19].
- Use **retrieval techniques**.
- Process the documents by **chunks** (e.g. summarize chapter by chapter).

What about other atchitectures?

Presentation H3.

Table of Contents

Traditional evaluation

Evaluating classification models is easily done with **accuracy, precision, recall, F1-score**.

But what about **generative models**?

Automatic traditional evaluation of generative models

BLEU [1] and ROUGE [2] metrics compare the **n-gram overlap** between a generated text and a reference one.

Reference	**Candidate**	**BLEU**
The quick brown fox jumps over the lazy dog.	The fast brown fox jumps over the sleeping dog.	47

Table 5: Examples of BLEU (as a rule of thumb BLEU is high when it's over 40).

Basically, the **more words in common, the higher they are**.

Automatic traditional evaluation of generative models

BLEU [1] and ROUGE [2] metrics compare the **n-gram overlap** between a generated text and a reference one.

Reference	**Candidate**	**BLEU**
The quick brown fox jumps over the lazy dog.	The fast brown fox jumps over the sleeping dog.	47
The weather is pleasant.	The climate is nice.	0
The conference room has a big problem.	The conference room has a large table.	54

Table 5: Examples of BLEU (as a rule of thumb BLEU is high when it's over 40).

Basically, the **more words in common, the higher they are**.

More accuracte evaluation

Problem: How can we capture semantic meaning more accurately?

Solution: Ask humans to rate the generated texts.

$\Longrightarrow$ Human preferences is still today the reference way to evaluate generative models.

Human based evaluation

The quality of a generative model is very often assessed through a **human evaluation**, on top of the automatic evaluation.

Human evaluation

Humans are asked to rate generated texts along several criteria:

- Fluency,
- Coherence,
- Relevance,
- Factuality,
- etc.

Human based evaluation

The quality of a generative model is very often assessed through a **human evaluation**, on top of the automatic evaluation.

Human evaluation

Humans are asked to rate generated texts along several criteria:

- Fluency,
- Coherence,
- Relevance,
- Factuality,
- etc.

But it is **time consuming** and **expensive**. In practice it is often performed on a reduced number of samples.

Model based evaluation

- Automatic n-gram based evaluation is **limited**.
- Human evaluation is **expensive**.

Model based evaluation

- Automatic n-gram based evaluation is **limited**.
- Human evaluation is **expensive**.

$\Longrightarrow$ Use LLMs instead!

Two kinds of model-based metrics

Models have been used in different ways:

- Embeddings-based.
- Human-like evaluation.

BERT Score

BERT Score [12] leverages contextualized information.

- Word embeddings are contextualized through a **BERT** model.
- The score is calculated based on the **cosine similarity** between the embeddings of words in the reference and generated texts.

$$R_{\text{BERT}} = \frac{1}{L} \sum_{i=1}^{L} \max_{1 \le j \le \hat{L}} \boldsymbol{y}_i^\top \hat{\boldsymbol{y}}_j,$$

$$P_{\text{BERT}} = \frac{1}{\hat{L}} \sum_{j=1}^{\hat{L}} \max_{1 \le i \le L} \boldsymbol{y}_i^\top \hat{\boldsymbol{y}}_j,$$

$$F_{\text{BERT}} = \frac{2 \cdot P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}.$$

Other model based metrics

Below are examples of other model-based metrics, where we extract some features from the model.

- BLEURT [10],
- COMET [8],
- BARTScore [9],
- etc.

GPT Evaluation

Since human evaluation is expensive, simply ask GPT4 to do it for you [18].

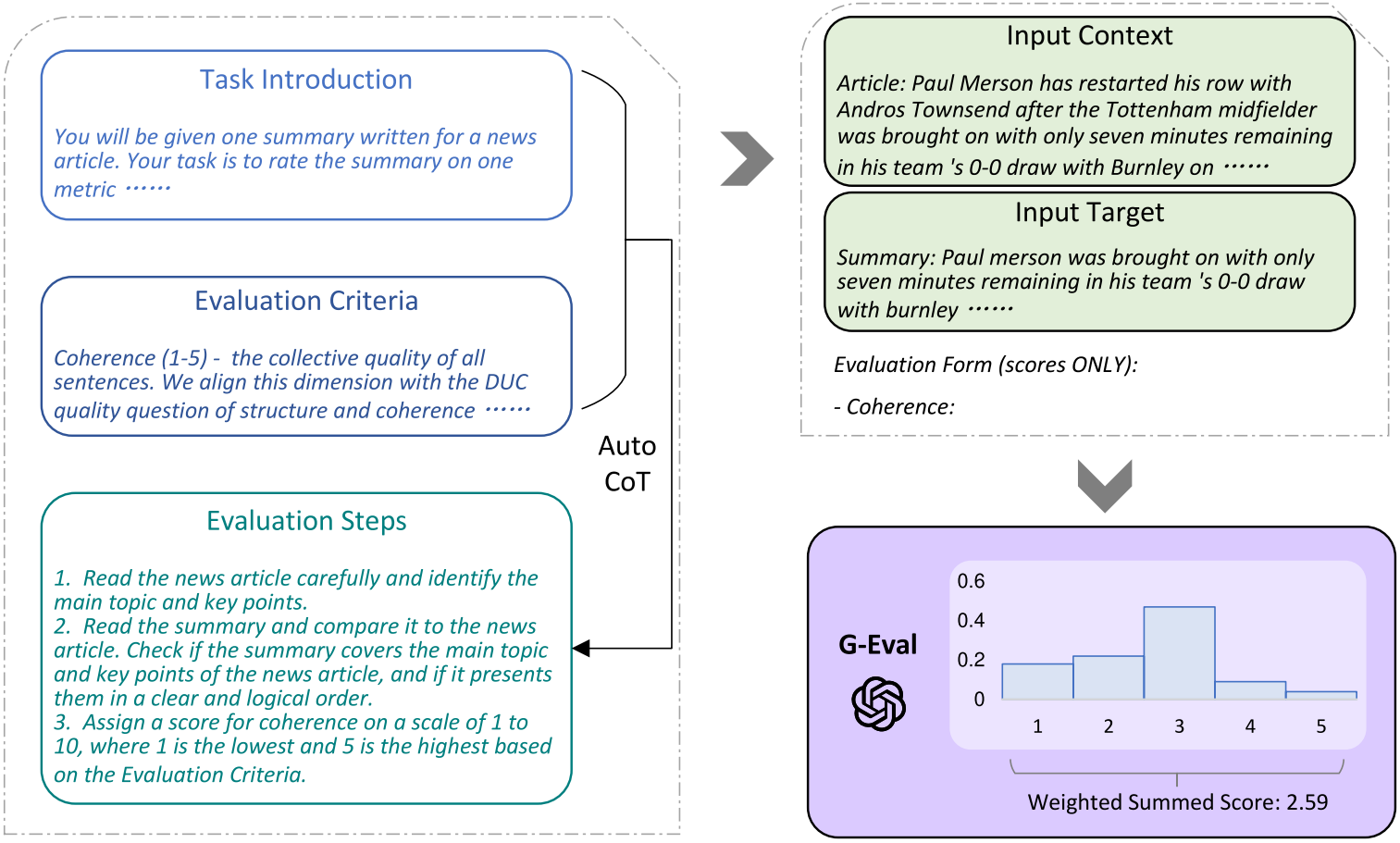


Figure 15: G-Eval, figure from [18].

Problems

Any thoughts?

Problems

Any thoughts?

A big blackbox model is evaluated by a big blackbox model.

LLM specific evaluation

In generalist LLM papers, there are often three kinds of evaluation:

- Human evaluation, performed on a limited number of samples,
- GPT4-Evaluation,
- Generalist benchmarks [13]. These benchmarks comprise a range of supervised tasks that are straightforward to assess, and the models are evaluated based on their performance in these tasks.

Biases and toxicity

Data are biased, then **models are biased**. How can we evaluate that models are not biased nor toxic?

- Test if potential unfair biases in coreferences resolution [3].
- Use simple classifiers for toxicity.

Biases and toxicity

Data are biased, then **models are biased**. How can we evaluate that models are not biased nor toxic?

- Test if potential unfair biases in coreferences resolution [3].
- Use simple classifiers for toxicity.

$\Longrightarrow$ There are **no actual guarantees** on such topics. Models like ChatGPT relies on empirical prompts and tricks to avoid toxicity and biases.

Table of Contents

Conclusion

Thank you!

References I

[1] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. 2002, pp. 311–318.

[2] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text summarization branches out: Proceedings of the ACL-04 workshop*. 2004, pp. 74–81.

[3] Rachel Rudinger et al. *Gender Bias in Coreference Resolution*. 2018. arXiv: 1804.09301 [cs.CL].

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[5] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].

References II

[6] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL].

[7] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[8] Mounica Maddela, Rico Sennrich, and Yvette Graham. *COMET: A Neural Framework for MT Evaluation*. 2020. arXiv: 2004.12867 [cs.CL].

[9] Feng Nan and Dan Klein. *BARTScore: Evaluating Generated Text as Text Generation*. 2020. arXiv: 2004.01970 [cs.CL].

[10] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. *BLEURT: Learning Robust Metrics for Text Generation*. 2020. arXiv: 2004.04696 [cs.CL].

[11] Mohammad Shoeybi et al. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. 2020. arXiv: 1909.08053 [cs.CL].

References III

[12] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL].

[13] Leo Gao et al. *A framework for few-shot language model evaluation*. Version v0.0.1. Sept. 2021. DOI: 10.5281/zenodo.5371628. URL: https://doi.org/10.5281/zenodo.5371628.

[14] Pengcheng He et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. 2021. arXiv: 2006.03654 [cs.CL].

[15] Manzil Zaheer et al. *Big Bird: Transformers for Longer Sequences*. 2021. arXiv: 2007.14062 [cs.LG].

[16] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. 2022. arXiv: 2205.14135 [cs.LG].

[17] Mandy Guo et al. *LongT5: Efficient Text-To-Text Transformer for Long Sequences*. 2022. arXiv: 2112.07916 [cs.CL].

[18] Yang Liu et al. *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. 2023. arXiv: 2303.16634 [cs.CL].

References IV

[19] Mathieu Ravaut et al. *On Position Bias in Summarization with Large Language Models*. 2023. arXiv: 2310.10570 [cs.CL].

[20] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].