

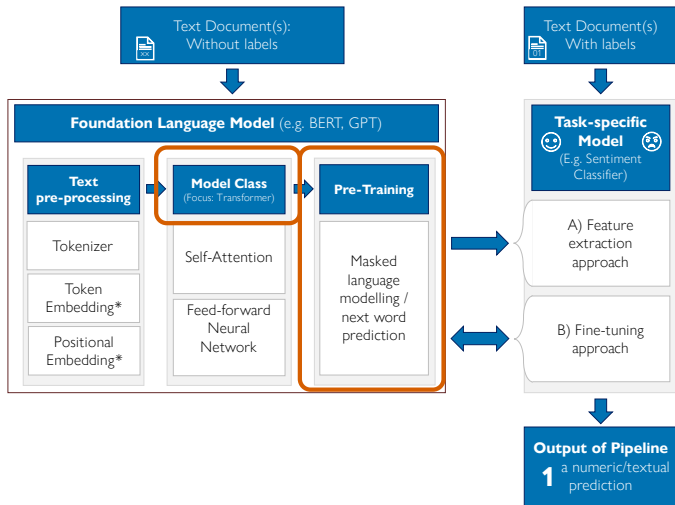
Understanding and Training Language Models: Training Foundation Language Models

Erik-Jan Senn

Faculty of Mathematics and Statistics, University of St. Gallen

CSH Autumn School at University of Hohenheim
September/October 2024

Where are we?



Language Modelling Pipeline

Foundation Models

Goal: Models that generally understand and predict human language well.

Foundation Models

Goal: Models that generally understand and predict human language well. **Conjecture:** If they are trained with sufficiently much data, they can work well for many possible tasks.

Foundation Models

Goal: Models that generally understand and predict human language well. **Conjecture:** If they are trained with sufficiently much data, they can work well for many possible tasks.

- ▶ **Non-generative / representation** transformers:
encoder-only models (e.g., *BERT*) aim to (only) find valuable latent text representations (but could also generate text).
- ▶ **Generative** learners:
Decoder-only transformers (e.g., *GPT*) aim to do autoregressive text generation (but also has valuable latent text representations).
- ▶ Generative **Encoder-decoder** transformers (e.g., *BART*) do both.

Roadmap

Training Foundation Models

Practical Aspects

Foundation Language Models

Training Foundation Models

Self-supervised Learning

1. **Issue:** We want to train a large model using supervised learning.
But how do we get the large amounts of data required for training?

Self-supervised Learning

1. **Issue:** We want to train a large model using supervised learning.
But how do we get the large amounts of data required for training?
2. **Idea:** Labels and the prediction target can be the text itself!
We can predict words from surrounding information, by masking words in the text.

Self-supervised Learning

1. **Issue:** We want to train a large model using supervised learning.
But how do we get the large amounts of data required for training?
2. **Idea:** Labels and the prediction target can be the text itself!
We can predict words from surrounding information, by masking words in the text.
3. Remaining setup:
 - ▶ Model class $f(\mathbf{x}, \theta)$: We use Transformers, which are easy to estimate and could work well for language. But can be any model.
 - ▶ Loss \mathcal{L} is a multi-class classification loss for the masked words.
 - ▶ The learning algorithm is e.g. gradient-based.

Self-supervised Learning

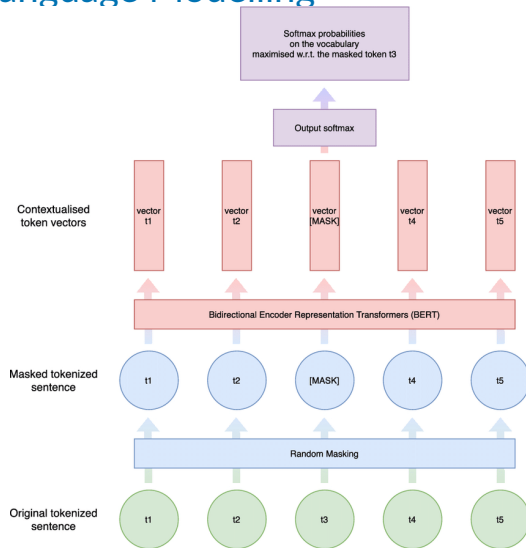
1. **Issue:** We want to train a large model using supervised learning.
But how do we get the large amounts of data required for training?
2. **Idea:** Labels and the prediction target can be the text itself!
We can predict words from surrounding information, by masking words in the text.
3. Remaining setup:
 - ▶ Model class $f(\mathbf{x}, \theta)$: We use Transformers, which are easy to estimate and could work well for language. But can be any model.
 - ▶ Loss \mathcal{L} is a multi-class classification loss for the masked words.
 - ▶ The learning algorithm is e.g. gradient-based.

Self-supervised Learning

1. **Issue:** We want to train a large model using supervised learning.
But how do we get the large amounts of data required for training?
2. **Idea:** Labels and the prediction target can be the text itself!
We can predict words from surrounding information, by masking words in the text.
3. Remaining setup:
 - ▶ Model class $f(\mathbf{x}, \theta)$: We use Transformers, which are easy to estimate and could work well for language. But can be any model.
 - ▶ Loss \mathcal{L} is a multi-class classification loss for the masked words.
 - ▶ The learning algorithm is e.g. gradient-based.

We turned a complex problem understanding human language into a multi-class prediction problem with lots a labeled data.

Random Masked Language Modelling



Random Masking in *BERT*. [Source](#)

Random Masked Language Modelling - Details

1. One-hot encoded input sequence $\mathbf{X} \in \mathbb{R}^{l \times v}$ (or id vector)

Random Masked Language Modelling - Details

1. One-hot encoded input sequence $\mathbf{X} \in \mathbb{R}^{l \times v}$ (or id vector)
2. 15% of input tokens are **randomly masked** (altered during training:
 - ▶ 80% out of are set to to the special token [Mask]. These will be our prediction targets later.
 - ▶ Some are replaced by a different token.

Random Masked Language Modelling - Details

1. One-hot encoded input sequence $\mathbf{X} \in \mathbb{R}^{l \times v}$ (or id vector)
2. 15% of input tokens are **randomly masked** (altered during training:
 - ▶ 80% out of are set to to the special token [Mask]. These will be our prediction targets later.
 - ▶ Some are replaced by a different token.
3. Input embeddings map to $\mathbf{E} \in \mathbb{R}^{l \times d}$
4. Transformer processes input embeddings to useful hidden representation $\mathbf{Z} \in \mathbb{R}^{l \times d}$ (keep dimension).

Random Masked Language Modelling - Details

1. One-hot encoded input sequence $\mathbf{X} \in \mathbb{R}^{l \times v}$ (or id vector)
2. 15% of input tokens are **randomly masked** (altered during training:
 - ▶ 80% out of are set to to the special token [Mask]. These will be our prediction targets later.
 - ▶ Some are replaced by a different token.
3. Input embeddings map to $\mathbf{E} \in \mathbb{R}^{l \times d}$
4. Transformer processes input embeddings to useful hidden representation $\mathbf{Z} \in \mathbb{R}^{l \times d}$ (keep dimension).
5. **Mask token prediction** for [Mask] tokens to the vocabulary v (excluding mask).
 - ▶ 1-layer MLP for classification (linear layer and softmax).
 - ▶ $\mathbf{Z} \mapsto \mathbf{P}$ where $\mathbf{P}_{\text{one hot}}$ with dimension $l \times v$.

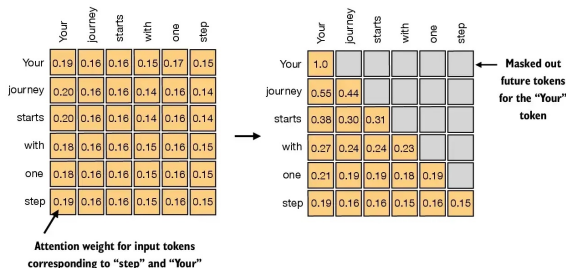
Random Masked Language Modelling - Details

1. One-hot encoded input sequence $\mathbf{X} \in \mathbb{R}^{l \times v}$ (or id vector)
2. 15% of input tokens are **randomly masked** (altered during training:
 - ▶ 80% out of are set to to the special token [Mask]. These will be our prediction targets later.
 - ▶ Some are replaced by a different token.
3. Input embeddings map to $\mathbf{E} \in \mathbb{R}^{l \times d}$
4. Transformer processes input embeddings to useful hidden representation $\mathbf{Z} \in \mathbb{R}^{l \times d}$ (keep dimension).
5. **Mask token prediction** for [Mask] tokens to the vocabulary v (excluding mask).
 - ▶ 1-layer MLP for classification (linear layer and softmax).
 - ▶ $\mathbf{Z} \mapsto \mathbf{P}$ where $\mathbf{P}_{\text{one hot}}$ with dimension $l \times v$.
6. In training, the loss propagates though the network, optimizing all trainable parameters!

Causal Masking

A **causal mask** within the **self-attention** mechanism avoids tokens from influencing tokens *in the future*.

- ▶ Key idea for (efficient) training of LMs.
- ▶ **Next-word prediction** uses a causal mask (decoders such as *GPT*). Bidirectional models use a random mask (e.g. encoders such as *BERT*).

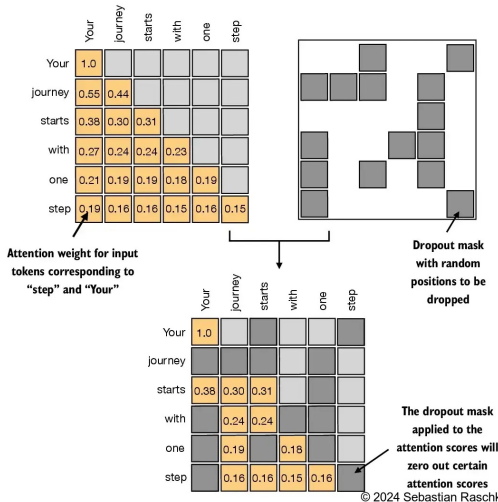


© 2024 Sebastian Raschka [Source](#)

Attention is normalized again row-by-row after masking.

Remark: Dropout for Self-attention

Dropout means randomly setting some outputs to 0 during training (regularization).



Source

Preparing Sequences for Next Word Prediction

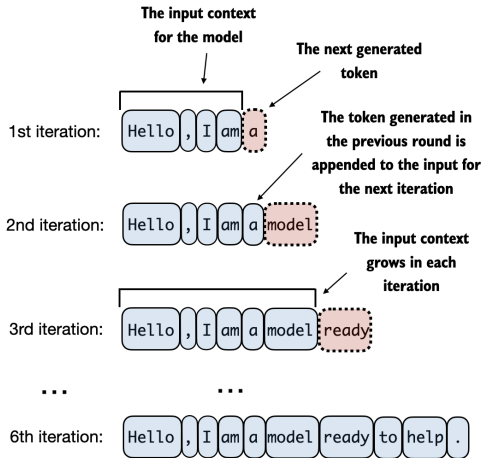


Figure adapted from "Build a Large Language Model From Scratch"

Source

Foundation Language Models

Practical Aspects

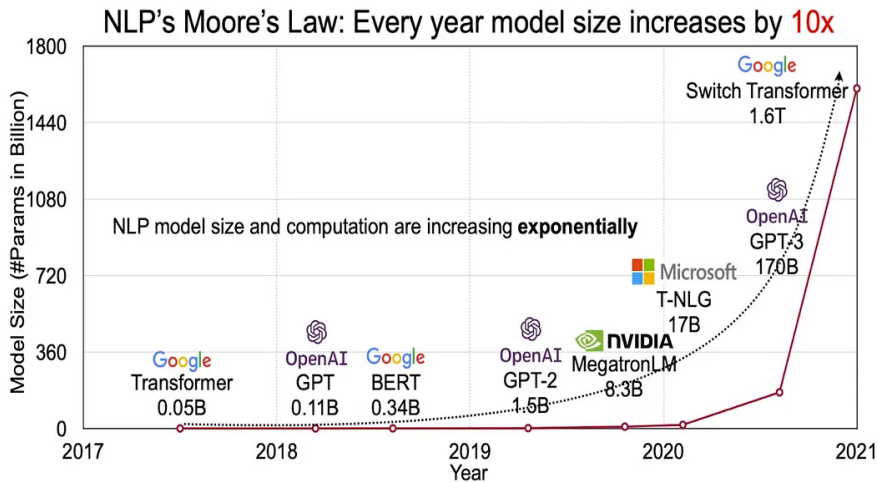
Training Data

Any open-source easily accessible text data available is collected.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

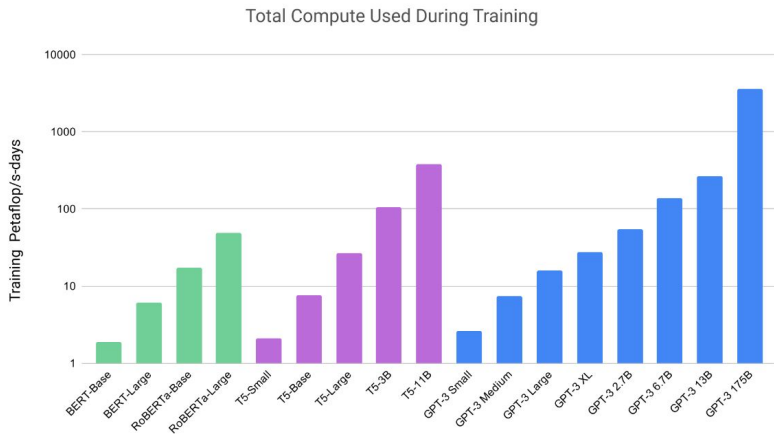
Example of GPT-3 training data, around 45 TB / 300 Billion tokens of text ([Brown et al., 2020](#)). Weight impacts the number of times a sample is seen in training.

Model Size is on the Rise



LLM parameters size over time [Source](#)

Computing Requirements and Power is on the Rise



Peta-Flop days used for Model Training ([Brown et al., 2020](#)). 1 Peta-flop day is approximately 3.2 days on a *NVIDIA A100*.

Computing and Memory Aspects of LMs

Computing steps:

1. Tokenization
2. Forward pass (computed in parallel for each token)
3. Backward pass (computed in parallel for each token)
4. Parameter update

Computing and Memory Aspects of LMs

Computing steps:

1. Tokenization
2. Forward pass (computed in parallel for each token)
3. Backward pass (computed in parallel for each token)
4. Parameter update

What elements are typically **stored on the GPU**?

Note: Example for *BERT*, to be tested empirically.

Computing and Memory Aspects of LMs

Computing steps:

1. Tokenization
2. Forward pass (computed in parallel for each token)
3. Backward pass (computed in parallel for each token)
4. Parameter update

What elements are typically **stored on the GPU**?

Note: Example for *BERT*, to be tested empirically.

- ▶ The **model**: p parameters `float32`: $110Mil \times 4 = 440MB$

Computing and Memory Aspects of LMs

Computing steps:

1. Tokenization
2. Forward pass (computed in parallel for each token)
3. Backward pass (computed in parallel for each token)
4. Parameter update

What elements are typically **stored on the GPU**?

Note: Example for *BERT*, to be tested empirically.

- ▶ The **model**: p parameters `float32`: $110Mil \times 4 = 440MB$
- ▶ The input data: $N \times l \times d$, where N is the batch size. For pretraining uses ($N = 256 \times l = 512$) = 128K tokens per batch.
 - ▶ **Data storage** : $256 \times 512 \times 768 \times 4bytes(float32) \approx 400.MB$ for input embedding.

Computing and Memory Aspects of LMs

Computing steps:

1. Tokenization
2. Forward pass (computed in parallel for each token)
3. Backward pass (computed in parallel for each token)
4. Parameter update

What elements are typically **stored on the GPU**?

Note: Example for *BERT*, to be tested empirically.

- ▶ The **model**: p parameters `float32`: $110Mil \times 4 = 440MB$
- ▶ The input data: $N \times l \times d$, where N is the batch size. For pretraining uses ($N = 256 \times l = 512$) = 128K tokens per batch.
 - ▶ **Data storage** : $256 \times 512 \times 768 \times 4\text{bytes}(\text{float32}) \approx 400.MB$ for input embedding.
 - ▶ Storing **activations** in the network for backpropagation scales
 - ▶ linearly in N
 - ▶ MLP-part linearly in l
 - ▶ Attention part **quadratically in l** .

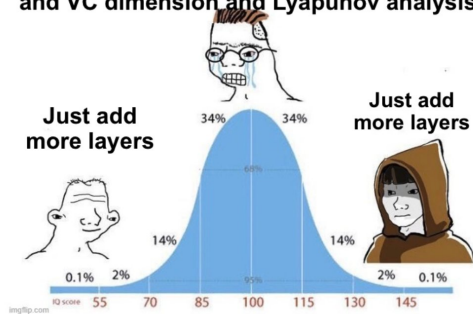
How did LLMs Improve in the Past?

- ▶ More **parameters** (transformer blocks, hidden dimensions, context size)
- ▶ More/better **data**

How did LLMs Improve in the Past?

- ▶ More **parameters** (transformer blocks, hidden dimensions, context size)
- ▶ More/better **data**
- ▶ Different **self-supervised tasks**:
 - ▶ Next sentence prediction
 - ▶ Permuted language modelling (predict word/sentence order in shuffled text)
- ▶ Not very important: **model architecture** details.
- ▶ Pretrained LM **performance comparison**: [Link Leaderboard](#), [Link Open LLM Leaderboard](#)

Nooooo you need convergence proofs and VC dimension and Lyapunov analysis



Source

Questions ?

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>.