Small Language Models

ACCELERATE PROGRAMME FOR SCIENTIFIC DISCOVERY

SCHMIDT FUTURES

Small language models

You may not actually need a high-parameter LLM

- Think about what you actually need and what task you are trying to do
  - Do you need chat/instruction capability?
  - Can your problem be framed in an alternative way?

- Example:
  - Gerry wants to look at how words are used in different contexts. His plan is to pick a word and get the dictionary definitions of that word.
  - He will then feed the dictionary definitions plus the sentences into an LLM and ask it to classify these sentences based on which definition the target word uses in the given context…

  … But this can be potentially reframed as sentence classification problem.

  - One approach would be to use a small language model, like BERT, to generate contextual embeddings, and use these embeddings as features in a classifier.

ACCELERATE PROGRAMME FOR SCIENTIFIC DISCOVERY

SCHMIDT FUTURES

How do you know which models to use?

How do we evaluate LLMs…?

How do you know which models to use?

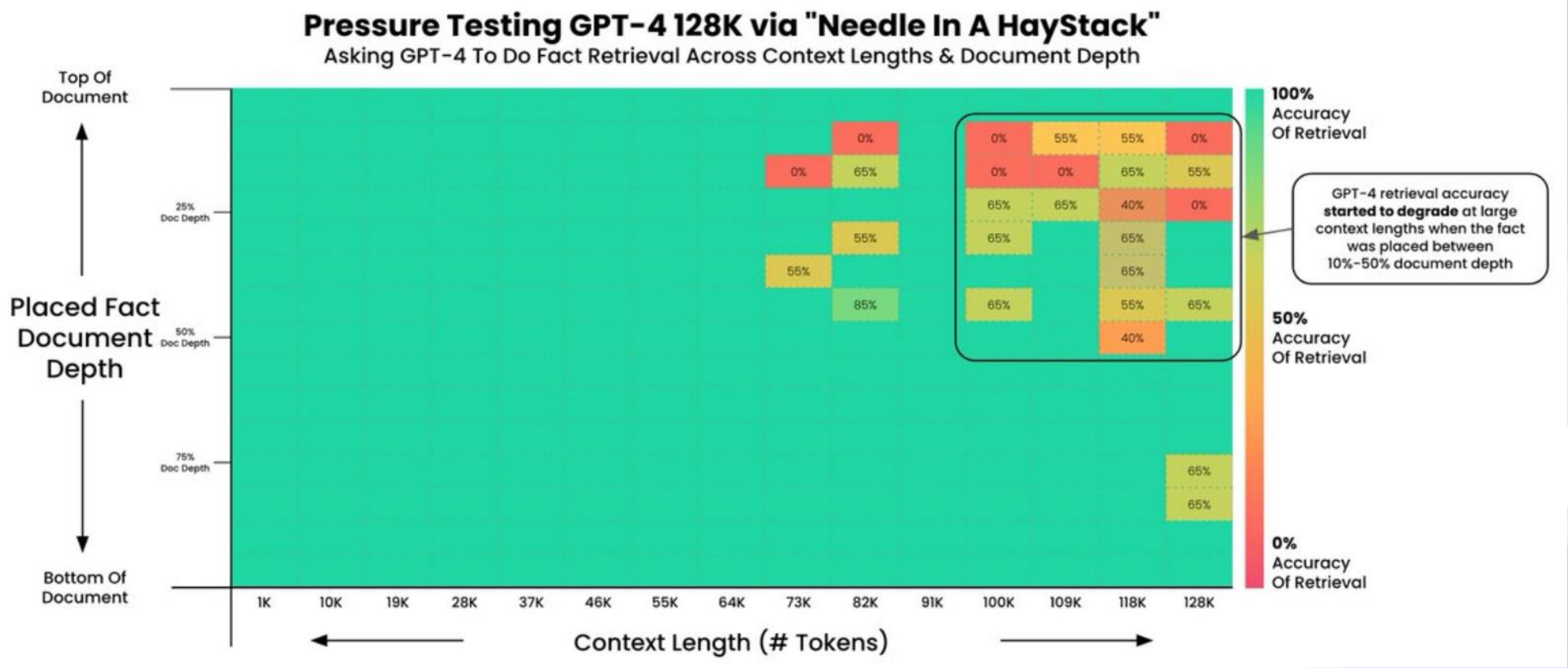
There is no magic rule that will tell you what size LLM you should choose. One place to look is on the the HuggingFace benchmarks

- MTEB Leaderboard
  - Measures a variety of text embedding tests such as semantic similarity
- Open LLM Leaderboard
  - Measures much more difficult tests
  - ARC, TruthfulQA, HellaSwag, etc.
- Find a task which maps onto a common benchmark.

Example

Needle in a haystack test

The first step is to decide what to work on. The work you choose needs to have three qualities: it has to be something you have a natural aptitude for, that you have a deep interest in, and that offers scope to do great work. The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day. In practice you don't have to worry much about the third criterion. Ambitious people are if anything already too conservative about it. So all you need to do is find something you have an aptitude for and great interest in.



Quantization

What is quantization and why is it important?

- Many of the most high-performing LLMs have approximately 70B parameters.
- In full precision, this takes up 140GiB of memory!
- A 32-bit floating point number has a precision of around 7 decimal places.
- But do we really need 7 decimal places…?
- Quantization is a technique of reducing the computational and memory costs of running inference by representing the weights of an LLM with a lower-precision data type.

ACCELERATE PROGRAMME FOR SCIENTIFIC DISCOVERY

SCHMIDT FUTURES

Quantization methods

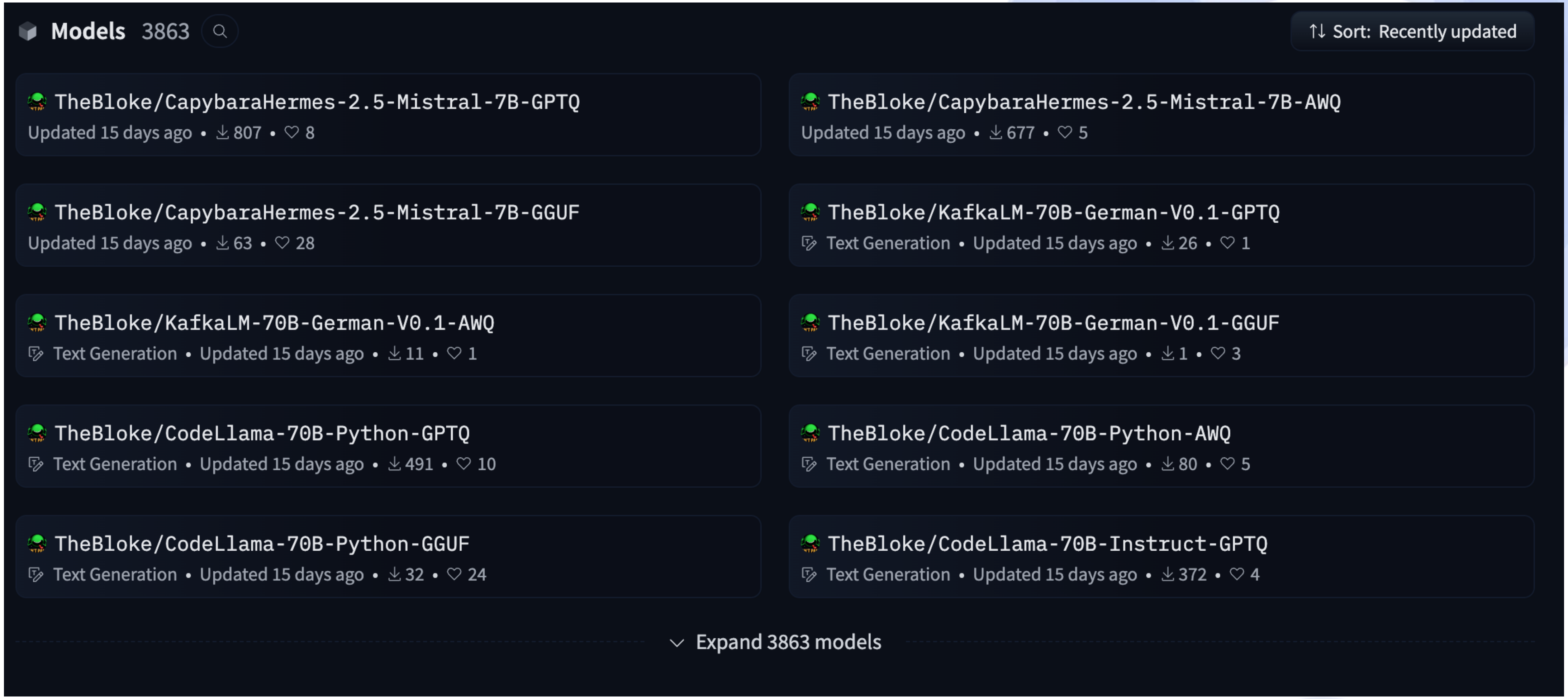
- Two types:
  - Post-training quantization (PTQ)
  - Quantization-aware training (QAT)

- PTQ methods:
  - GPTQ, AWQ
  - GGML, GGUF, and Llama.cpp
  - NF4
  - GGUF for Apple devices
  - GPTQ and AWQ for Nvidia devices

- Trade-off: should you use a model with few parameters, or use a model with more parameters, but that has been quantized?

- PEFT and BitsAndBytes libraries.

Where to find quantized models.



Models 3863

Sort: Recently updated

TheBloke/CapybaraHermes-2.5-Mistral-7B-GPTQ
Updated 15 days ago • 807 • 8

TheBloke/CapybaraHermes-2.5-Mistral-7B-AWQ
Updated 15 days ago • 677 • 5

TheBloke/CapybaraHermes-2.5-Mistral-7B-GGUF
Updated 15 days ago • 63 • 28

TheBloke/KafkaLM-70B-German-V0.1-GPTQ
Text Generation • Updated 15 days ago • 26 • 1

TheBloke/KafkaLM-70B-German-V0.1-AWQ
Text Generation • Updated 15 days ago • 11 • 1

TheBloke/KafkaLM-70B-German-V0.1-GGUF
Text Generation • Updated 15 days ago • 1 • 3

TheBloke/CodeLlama-70B-Python-GPTQ
Text Generation • Updated 15 days ago • 491 • 10

TheBloke/CodeLlama-70B-Python-AWQ
Text Generation • Updated 15 days ago • 80 • 5

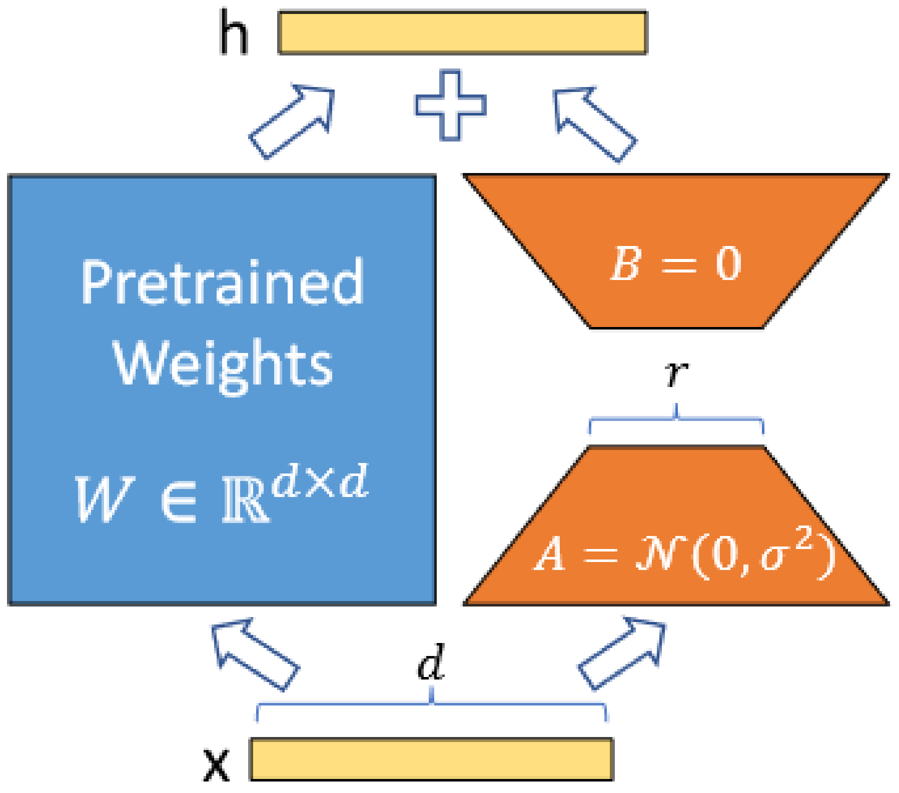
TheBloke/CodeLlama-70B-Python-GGUF
Text Generation • Updated 15 days ago • 32 • 24

TheBloke/CodeLlama-70B-Instruct-GPTQ
Text Generation • Updated 15 days ago • 372 • 4

Expand 3863 models

QLoRA

- Finetuning a model can be expensive!
- The new weights are denoted by
  $$W = W_0 + \Delta W$$
- The difference between the pretrained weights and finetuned weights, ΔW, is small and has low rank.
- Approximate ΔW by two low rank matrices instead.
- Each attention module has 4 matrices to train, and the final layers are usually fixed.
- Massively decreases the training cost, with minimal loss of performance.


h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

QLoRA

- In QLoRA, we can take advantage of quantization ***and*** LoRA!
- First, obtain the low rank decomposition of the weights ***and keep them in high precision***.
- Then quantize the original weight matrix.
- Train the network as with LoRA.
- The original weight matrix must be dequantized during the forward pass.
- Significant memory reductions, but with no significant degradation in performance!
- Only good for slightly guiding the model – you cannot instil significant new knowledge.

Resources

Intro to Quantization

HuggingFace OpenLLM Leaderboard

HuggingFace MTEB Leaderboard

TheBloke

QLoRA

PEFT