

IFC-Bank of Italy Workshop on "Data science in central banking"

18-20 February 2025

Tailoring small language models for enterprise use cases¹

Julien Simon,
Arcee.ai

¹ This contribution was prepared for the workshop. The views expressed in this publication are those of the authors and do not necessarily represent the official views of the Committee, its members, or the BIS.



Tailoring small language models for enterprise use cases

Julien Simon, Chief Evangelist

julien@arcee.ai

[linkedin.com/in/juliensimon](https://www.linkedin.com/in/juliensimon)

[youtube.com/juliensimonfr](https://www.youtube.com/juliensimonfr)

Hard truths, part 1

If GenAI looks deceptively simple, then I'm afraid you are deceived

- How **deep** is the model's **knowledge** on your business domain?
- Not just your industry, but also your company, your department, your customers, etc.
- How **fresh** is that **knowledge**?
- How **compliant** are the **answers**? Structure, length, tone of voice, safety
- Does the model make it easy to **trust** its **answers**? Step by step "reasoning", quotes
- Prompt tuning is mostly a bad joke 🙄
- RAG is mandatory for fresh data, but doesn't deliver domain adaptation
- And what about privacy, security, compliance, cost, scaling, etc.



Hard truths, part 2

The « best » closed LLM is not all you need

- Closed LLMs have been built to be a **mile wide** and an **inch deep**
- They're convenient for **vanilla tasks** but quickly show their limits
- They don't know what they don't know and they can't be efficiently **fine-tuned**
« *How well do commercial fine-tuning APIs infuse knowledge into LLMs?* » <https://arxiv.org/abs/2409.05059> (11/2024)
- Users try to fix this with **huge-context RAG**, which doesn't work well, if at all
« *Lost in the Middle: How Language Models Use Long Contexts* » <https://arxiv.org/abs/2307.03172> (06/2023)
- You have **very few options** to tweak cost/business performance/technical performance
- Many business processes require not only a model, but possibly **several models**, and often **external IT tools** (documents, APIs, SQL, etc.)



Hard truths, part 3

The « best » open-source SLM is not all you need

- Yes, open-source SLMs solve most, **maybe all**, the closed LLM problems.
- There are still **challenges** to overcome (we'll talk about them in a minute)
- Last night, while you were sleeping, somebody released a **new** « best » SLM 🤖
- Many business processes require not only a model, but possibly **several models**, and often **external IT tools** (documents, APIs, SQL, etc.)



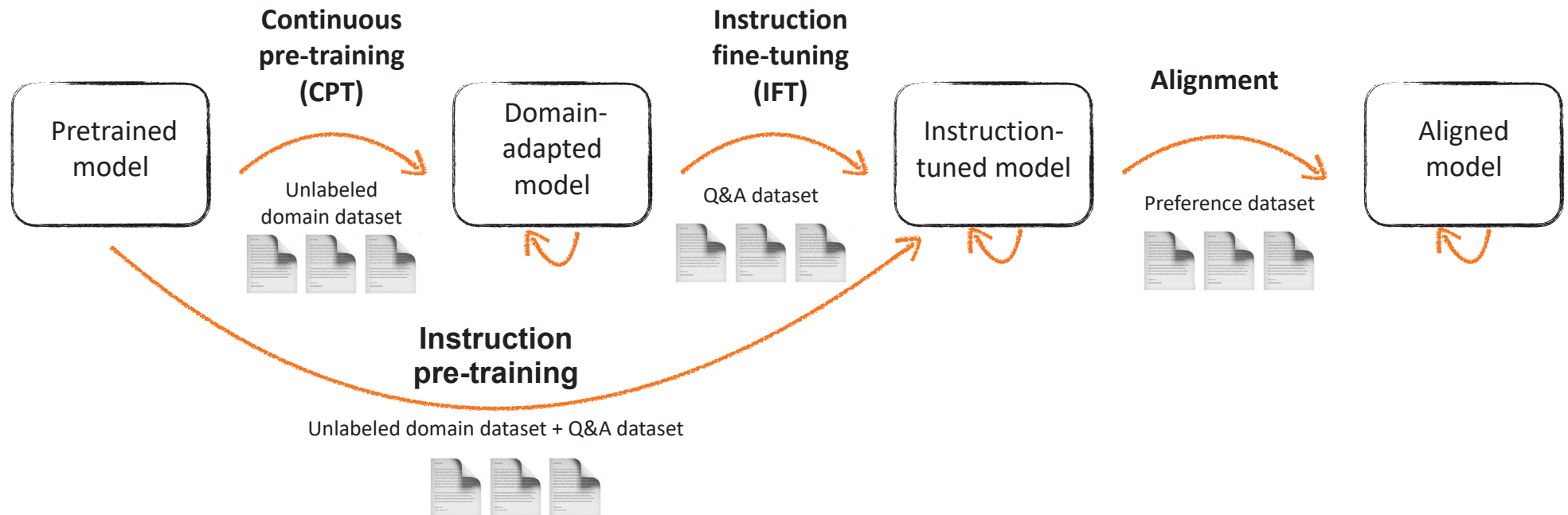
Our mission at Arcee AI

- Build **SLMs that consistently outperform** not only other SLMs, but also LLMs
- Build them **quickly and cost-efficiently** based on the « best » current architectures
- If need be, make it possible for you to **tailor** our SLMs
- Make it easy for you to **consume models** (ours and others)
 - Directly with **simple APIs** a la OpenAPI - Arcee Model Engine
 - In our **agentic workflow platform** - Arcee Orchestra
 - In our **next-gen router-based inference platform** - Arcee Conductor
- Give you several deployment options: **SaaS**, **Cloud** (VPC), **on-prem**
- Deliver **industry-leading GenAI solutions**, both on **quality** and **ROI**

How do we do it?



A typical SLM adaptation workflow



- « Language Models are Few-Shot Learners » <https://arxiv.org/abs/2005.14165> (05/2020)
- « Finetuned Language Models Are Zero-Shot Learners » <https://arxiv.org/abs/2109.01652> (09/2021)
- « Efficient Continual Pre-training for Building Domain Specific Large Language Models » <https://arxiv.org/abs/2311.08545> (11/2023)
- « Instruction Pre-Training: Language Models are Supervised Multitask Learners » <https://arxiv.org/abs/2406.14491v1> (06/2024)
- « How Do Large Language Models Acquire Factual Knowledge During Pretraining? » <https://arxiv.org/abs/2406.11813v1> (06/2024)



Challenges of SLM adaptation

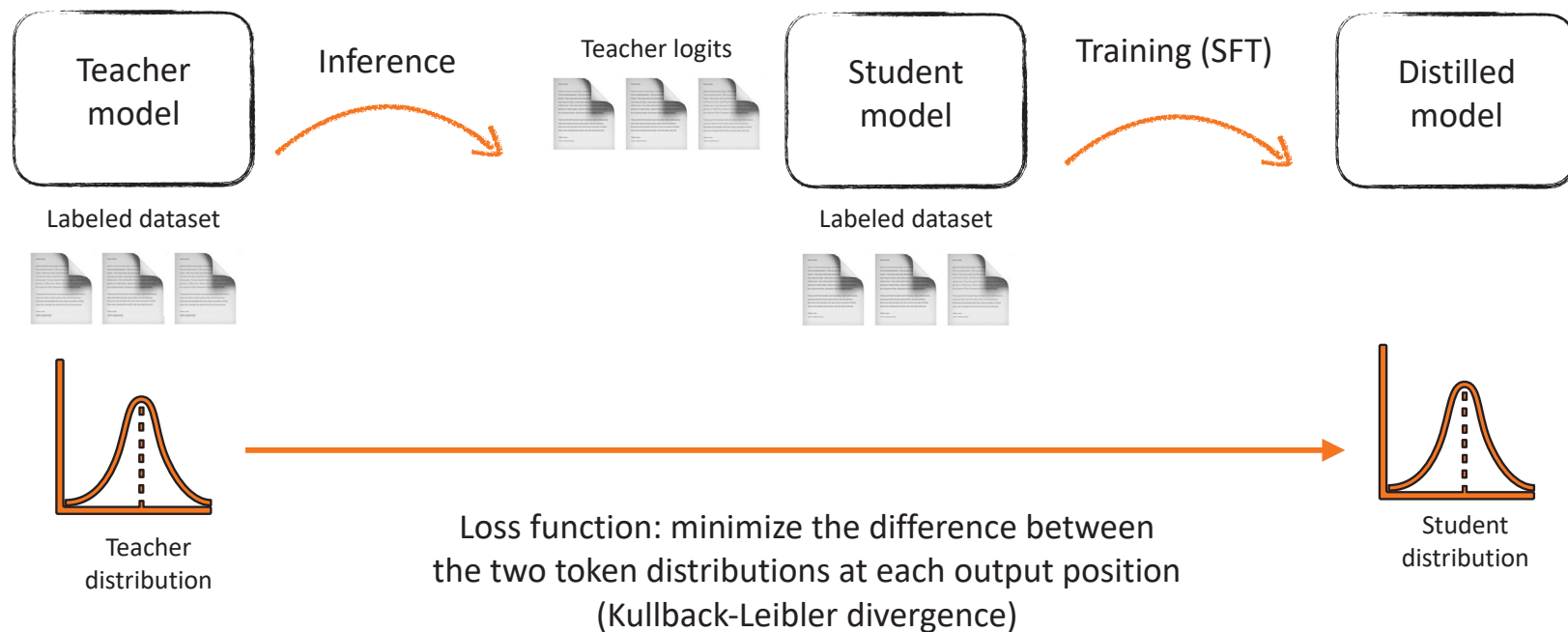
- Building datasets is hard work
 - Continuous Pre-Training requires a **large corpus** (at least 1 billion of tokens)
 - Instruction Fine-Tuning and Alignment requires **high-quality, diverse Q&A pairs**
- Training / fine-tuning models: accuracy or cost, pick one
 - **Working with the full model in original precision** (say, BF16)
 - Compute-heavy and expensive... assuming you can even get the amount of compute you need
 - **Parameter Efficient Fine Tuning** (PEFT), e.g. LoRA or QLoRA
 - Much more memory-efficient, enabling smaller GPUs and shorter training times
 - Effective for Instruction Fine-Tuning and Alignment
 - Significant accuracy degradation for CPT
 - <https://blog.arcee.ai/why-methods-like-qlora-fall-short-in-domain-knowledge-injection-2/>
 - « *LoRA vs. Full Fine-Tuning: An Illusion of Equivalence* » <https://arxiv.org/abs/2410.21228> (10/2024)
- Can we get FFT-like quality at a fraction of the cost? **Yes**



Model distillation: DistillKit

<https://blog.arcee.ai/distillkit-v0-1-by-arcee-ai/> (08/2024) + <https://github.com/arcee-ai/DistillKit>

Intuition: if we have excellent large open-source models available,
can we teach smaller models to mimic their output?



Parameter-efficient training: Spectrum

<https://arxiv.org/abs/2406.06623> (06/2024)+ <https://github.com/cognitivecomputations/spectrum>

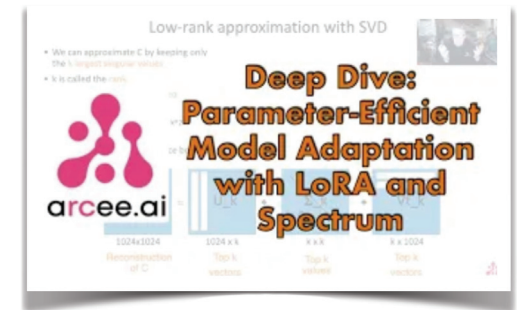
- Intuition: not all layers contribute equally to the output.
- Some layers have a higher **signal-to-noise ratio** (SNR) than others.
- Spectrum identifies the high SNR layers.
- Run **full fine-tuning** on high SNR layers, and leaves the other layers untouched.

$$SNR = \frac{\sum_{k|\sigma_k > \epsilon} \sigma_k}{\sum_{n|\sigma_n < \epsilon} \sigma_n}$$

```
python spectrum.py --model-name <insert local or HF repo here>
                  --top-percent <top % of SNR ratios to target>
```

```
"model.layers.10.self_attn.o_proj": {
  "snr": 0.25031203031539917,
  "type": "self_attn.o_proj"
},
"model.layers.11.self_attn.o_proj": {
  "snr": 0.2547757625579834,
  "type": "self_attn.o_proj"
},
"model.layers.12.self_attn.o_proj": {
  "snr": 0.2616233825683594,
  "type": "self_attn.o_proj"
},
"model.layers.13.self_attn.o_proj": {
  "snr": 0.2736438810825348,
  "type": "self_attn.o_proj"
},
. . .
```

```
unfrozen_parameters:
- ^lm_head.weight$
- ^model.embed_tokens.weight$
# input_layernorm layers
- model.layers.0.input_layernorm
- model.layers.1.input_layernorm
- model.layers.2.input_layernorm
- model.layers.3.input_layernorm
- model.layers.4.input_layernorm
- model.layers.5.input_layernorm
# lm_head layers
# mlp.down_proj layers
- model.layers.21.mlp.down_proj
- model.layers.20.mlp.down_proj
- model.layers.22.mlp.down_proj
- model.layers.19.mlp.down_proj
- model.layers.23.mlp.down_proj
- model.layers.24.mlp.down_proj
. . .
```

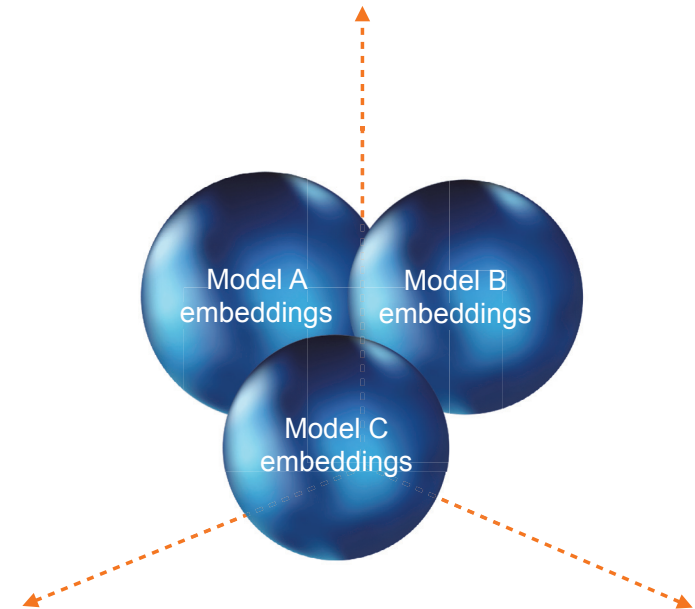
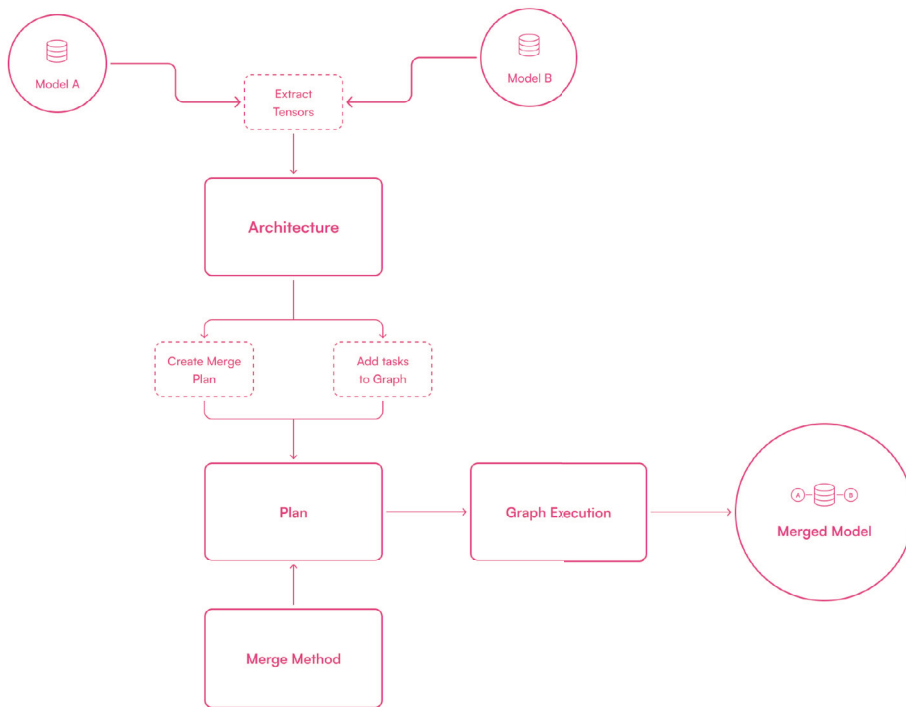


Model merging: MergeKit

<https://arxiv.org/abs/2403.13257> (03/2024) + <https://github.com/arcee-ai/mergekit>

Intuition: if we have plenty of open-source models, can we build a model by merging several models that already have the properties we need?

- Combine **several task-specific models** (or different checkpoints) into a single model with zero additional training
- Not an ensembling technique: only one model
- Merging only requires **lightweight compute**
- Fast process, no extra inference latency



Demo: merging models with mergekit

```
$ git clone https://github.com/arcee-ai/mergekit.git
$ cd mergekit
$ pip install -e .
```

```
models:
- model: defog/llama-3-sqlcoder-8b
  parameters:
    density: 1.0
    weight: 0.2
- model: MathGenie/MathCoder2-Llama-3-8B
  parameters:
    density: 1.0
    weight: 0.6
- model: ajibawa-2023/Code-Llama-3-8B
  parameters:
    density: 1.0
    weight: 0.2
merge_method: ties
base_model: meta-llama/Llama-3.1-8B
dtype: float16
```

72.48%

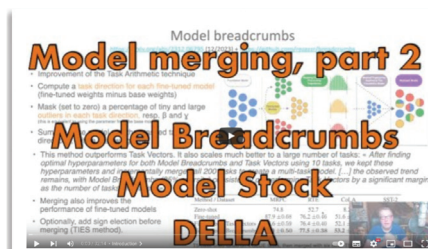
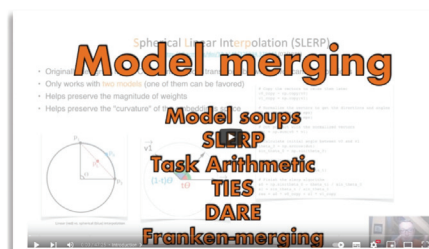
65.35%

64.37%

```
(env-mergekit) → examples git:(main) x mergekit-yaml ties-demo.yml ties-demo
Fetching 11 files: 100%|██████████| 11/11 [00:00<00:00, 19886.79it/s]
Fetching 11 files: 100%|██████████| 11/11 [00:00<00:00, 14377.48it/s]
Fetching 9 files: 100%|██████████| 9/9 [00:00<00:00, 10960.72it/s]
Fetching 10 files: 100%|██████████| 10/10 [00:00<00:00, 25070.56it/s]
Warmup loader cache: 100%|██████████| 4/4 [00:00<00:00, 5.05it/s]
Executing graph: 100%|██████████| 2039/2039 [01:05<00:00, 30.95it/s]
```

Merging three 8B models with TIES
1 minute on my Mac :)

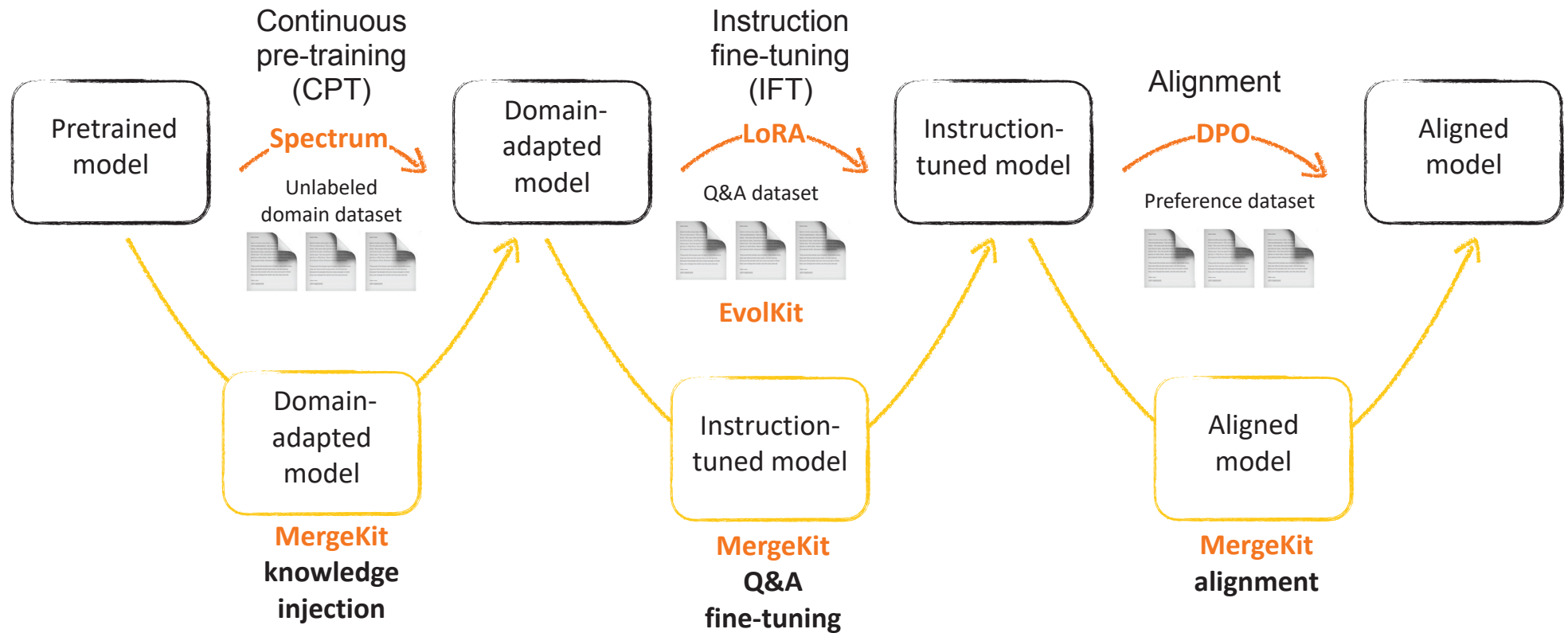
73.62%



GSM8K scores - lm_eval 12/16/2024



A modern SLM adaptation workflow



« Arcee's MergeKit: A Toolkit for Merging Large Language Models » <https://arxiv.org/abs/2403.13257> (03/2024)

« Spectrum: Targeted Training on Signal to Noise Ratio » <https://arxiv.org/abs/2406.06623> (06/2024)

« Merging in a Bottle: Differentiable Adaptive Merging (DAM) and the Path from Averaging to Automation » <https://arxiv.org/abs/2410.08371> (10/2024)



Arcee AI - The Open SLM leader

State-of-the-art tech stack based on open-source libraries

Spectrum (continuous pre-training), MergeKit (merging), DistilKit (distillation), EvolKit (dataset improvement)

Best-in-class models based on open-source architectures



Qwen2 1.5B



Best 1.5B model



Llama 3.1 8B



Best 8B model



Qwen2.5 14B



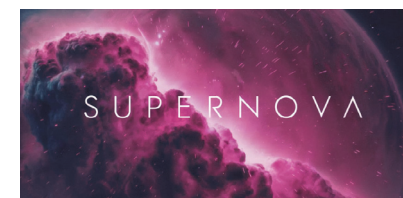
Best 14B model



Qwen2 72B



Best Arabic model



Llama 3.1 70B

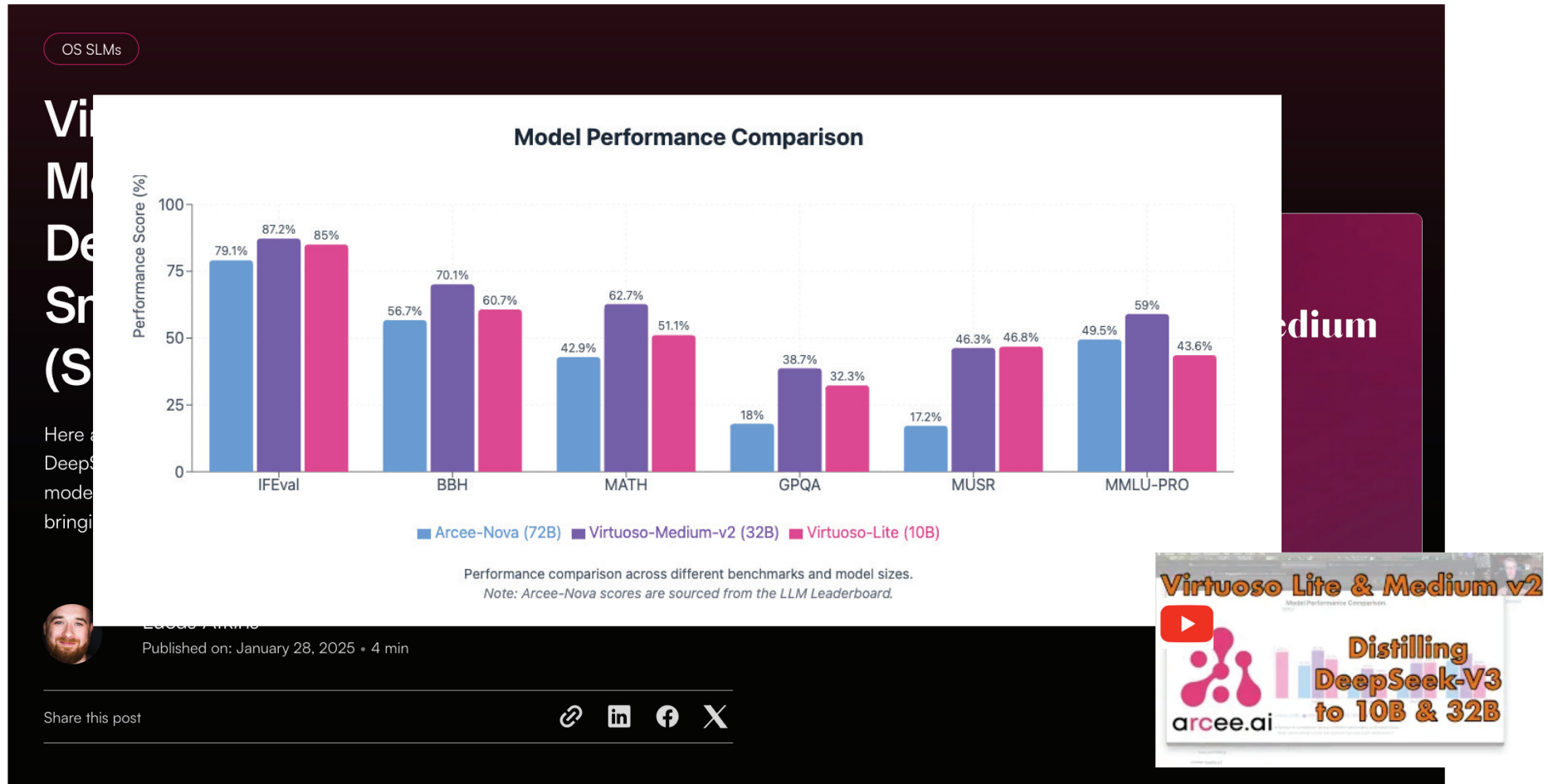


Best 70B model



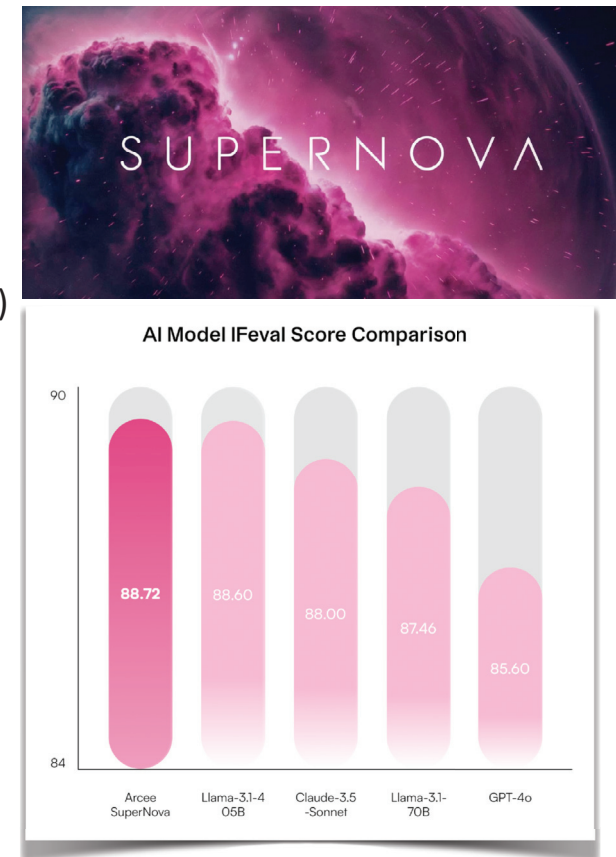
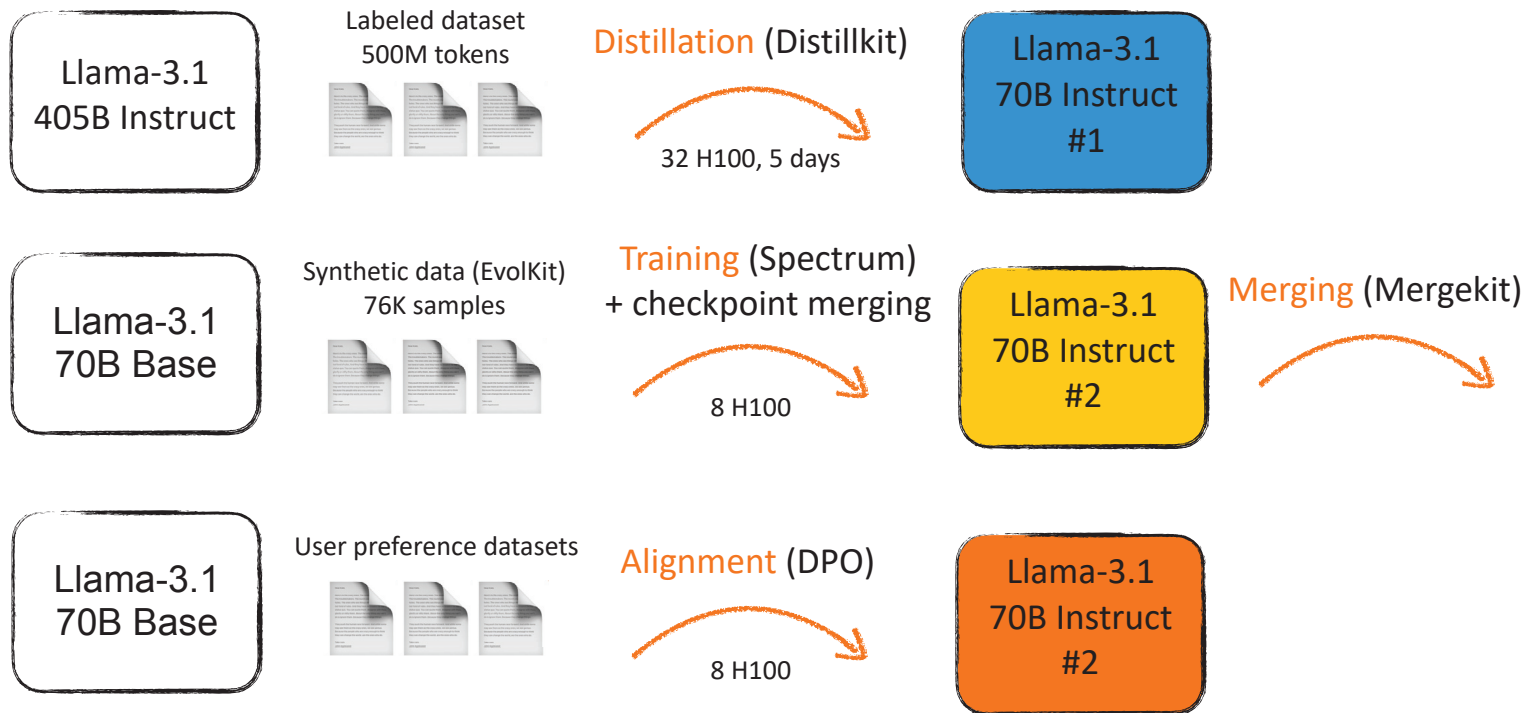
Latest releases: 10B and 32B

<https://www.arcee.ai/blog/virtuoso-lite-virtuoso-medium-v2-distilling-deepseek-v3-into-10b-32b-small-language-models-slms>



Arcee SuperNova training pipeline

<https://blog.arcee.ai/arcee-supernova-training-pipeline-and-model-composition/>



Arcee Model Engine - SLM inference platform

<https://www.arcee.ai/product/model-engine>

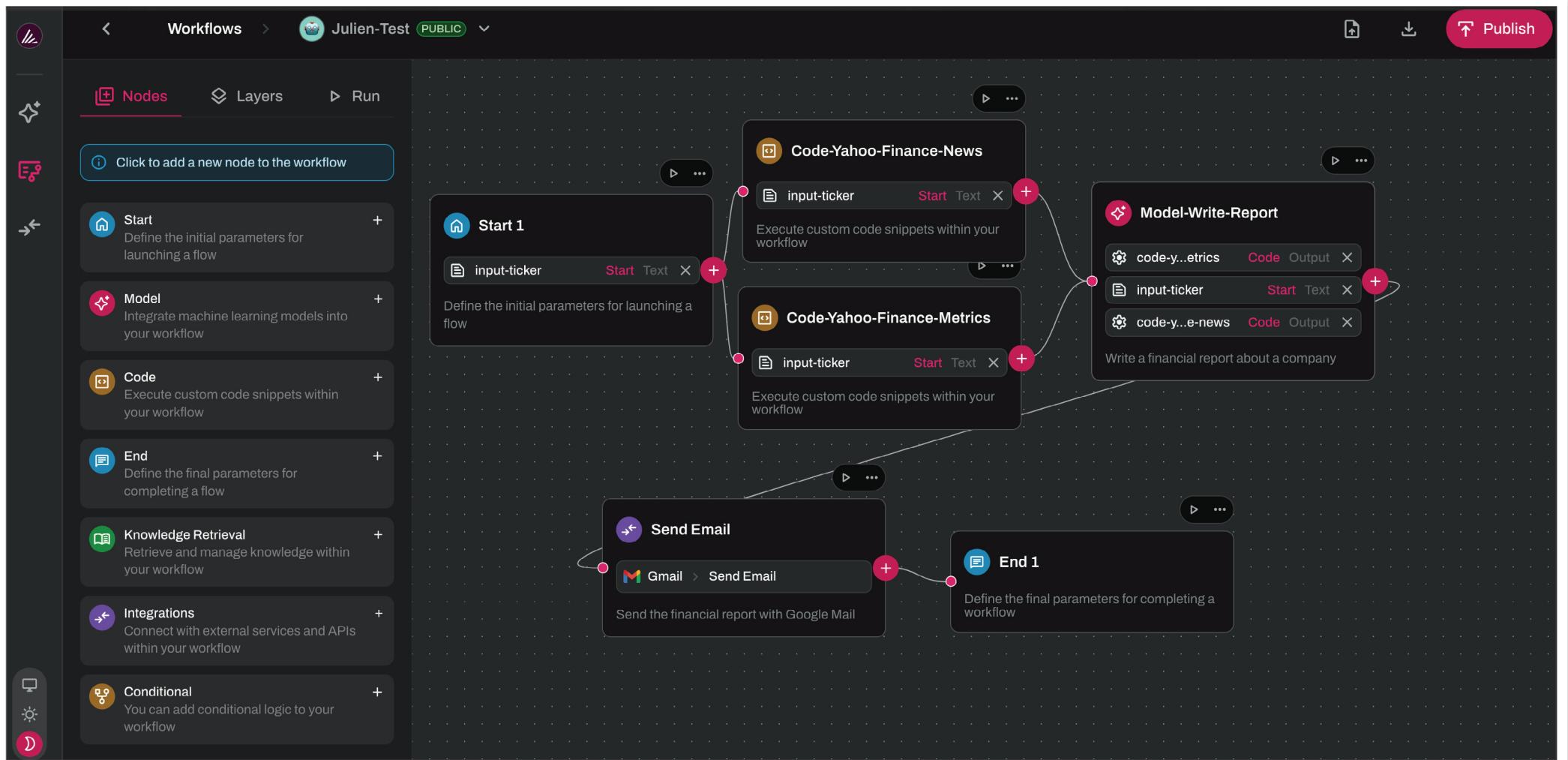
The screenshot displays the Arcee Model Engine dashboard. On the left is a sidebar with the Arcee logo and navigation links: Models, Payments, API Keys, Usage, and Contact. The main area features six model cards arranged in a 3x2 grid. Each card includes a model name and a brief description. A user profile icon with the letter 'J' is in the top right corner. At the bottom left are refresh and back navigation icons.

Model Name	Description
Virtuoso Large	Our most powerful and versatile general-purpose model
Virtuoso Medium	Versatile model capable of handling complex and varied tasks
Virtuoso Small	Streamlined version of the larger Virtuosos, faster & more cost-efficient
Coder Large	High-performance model tailored for intricate programming tasks
Coder Small	Compact coding model for efficient programming tasks
Caller Large	Optimized for complex tool-based interactions & API function calls



Arcee Orchestra - SLM agentic workflows

<https://www.arcee.ai/product/orchestra>



Arcee AI



Leader in Small
Language Models
(SLMs)

End-to-end
Training and Agentic
Platforms

SaaS and Private
Deployments

SAMSUNG



EMORY
HEALTHCARE

PIMCO



PROGRESSIVE



Julien Simon, Chief Evangelist

julien@arcee.ai

linkedin.com/in/juliensimon

youtube.com/juliensimonfr

