

# MindNLP：高性能极简易 用的自然语言处理套件

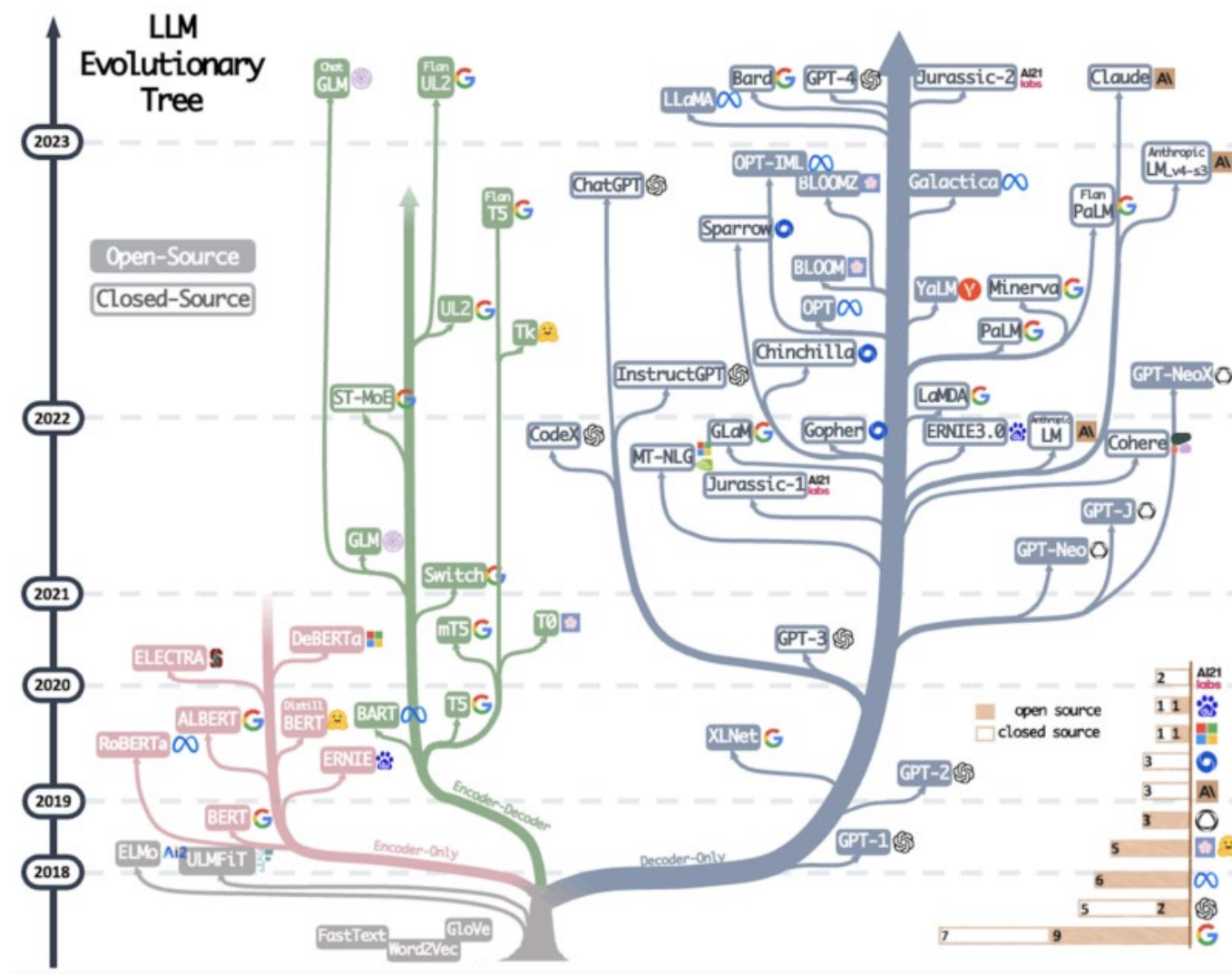
吕昱峰

MindSpore架构师/MindNLP负责人

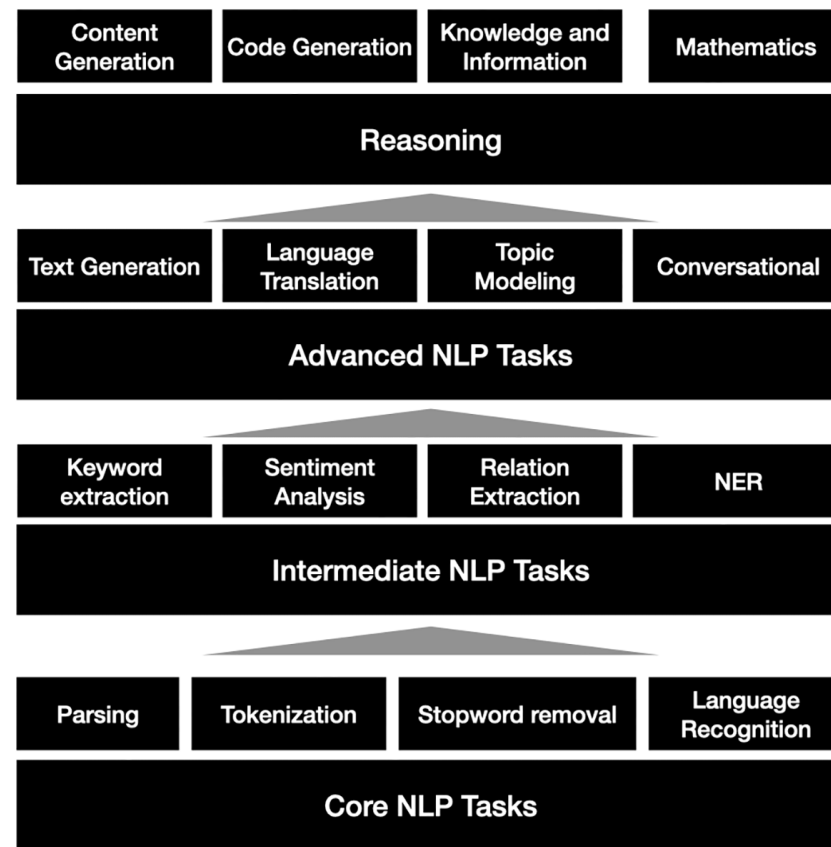
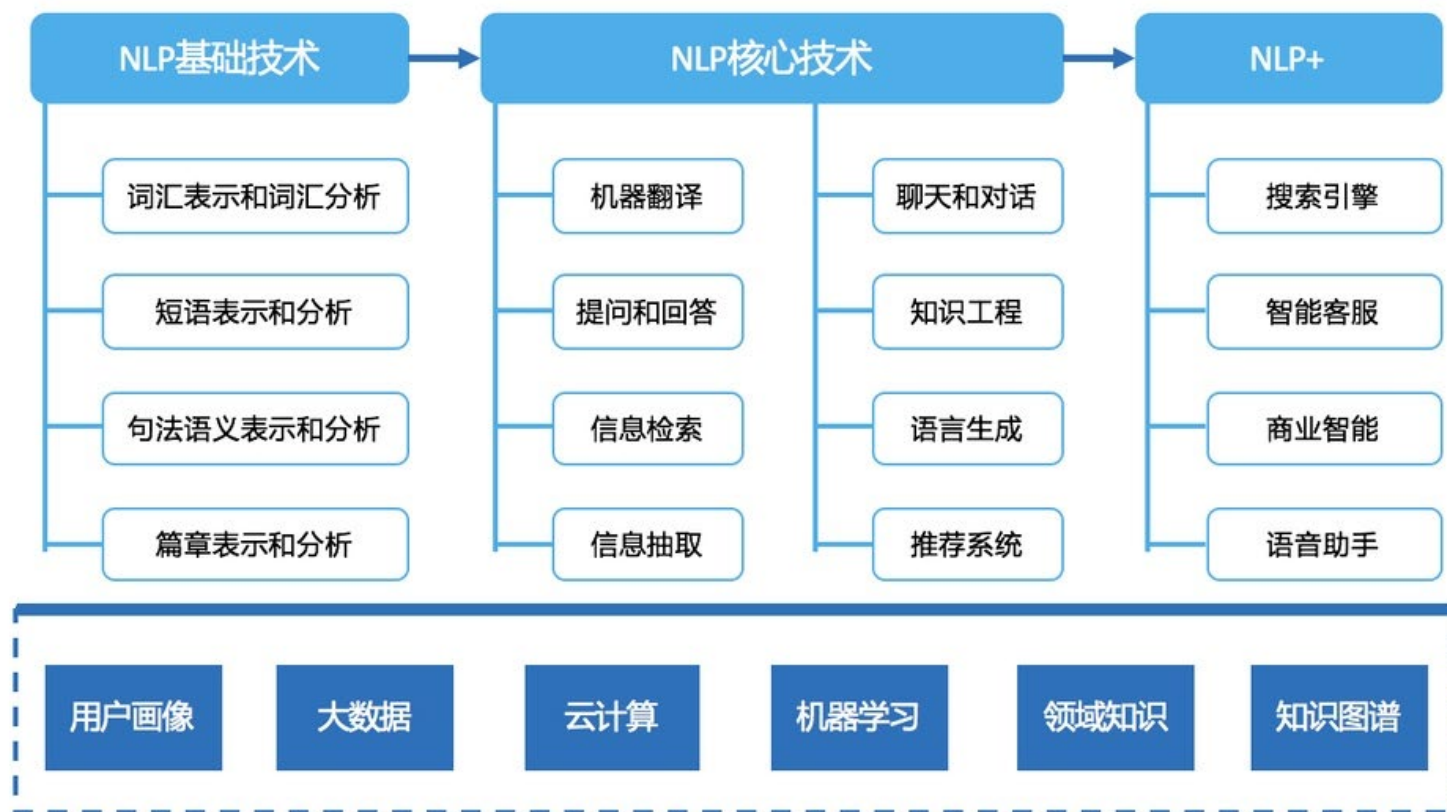
# 目录

- 为什么需要NLP套件
- 基于MindSpore优势特性的NLP套件设计
- 嫁接 🤗 生态——站在巨人的肩膀上
- MindNLP训推案例
- 高校同学的开源贡献

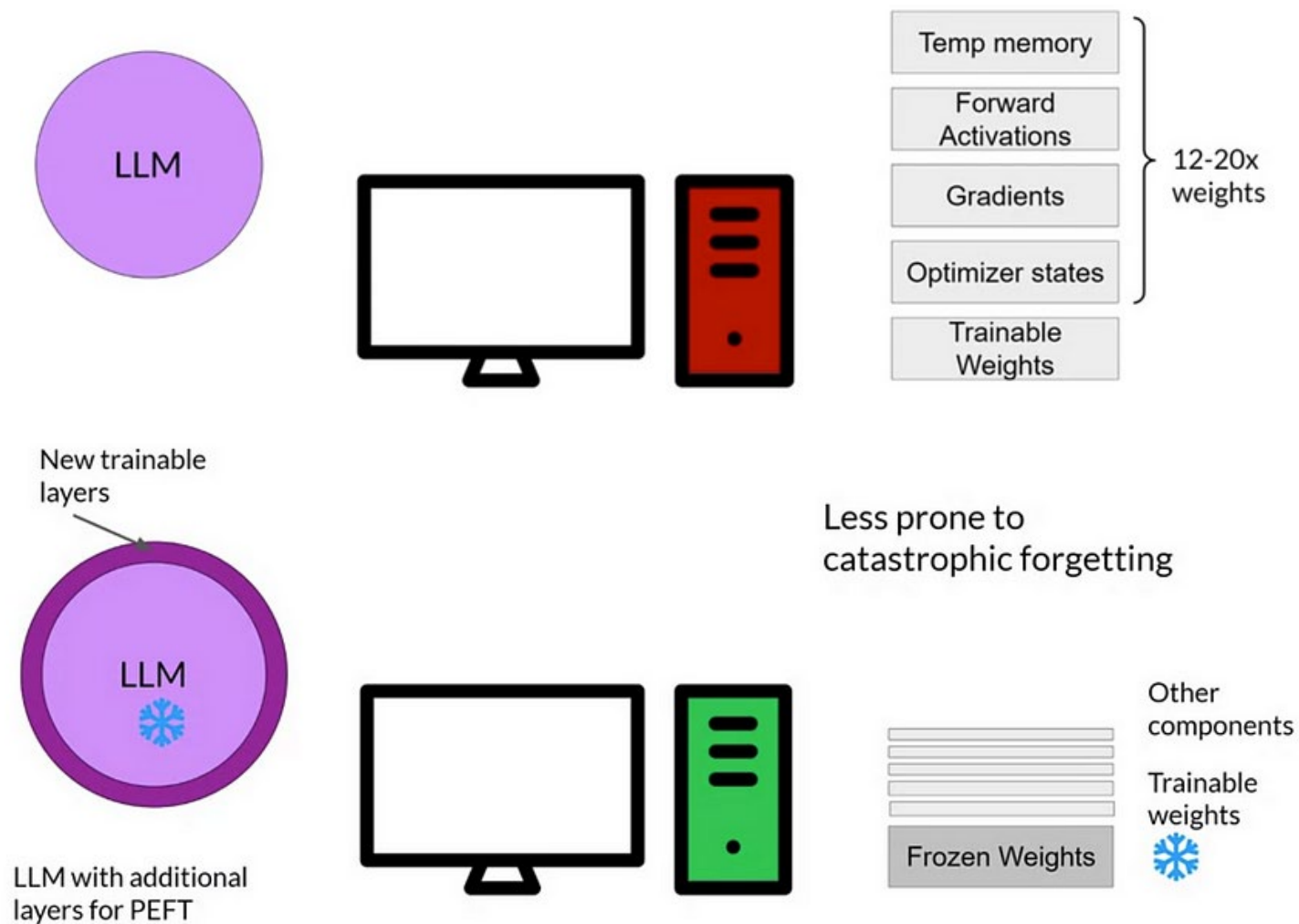
# 基于Transformer的LLM统一NLP范式



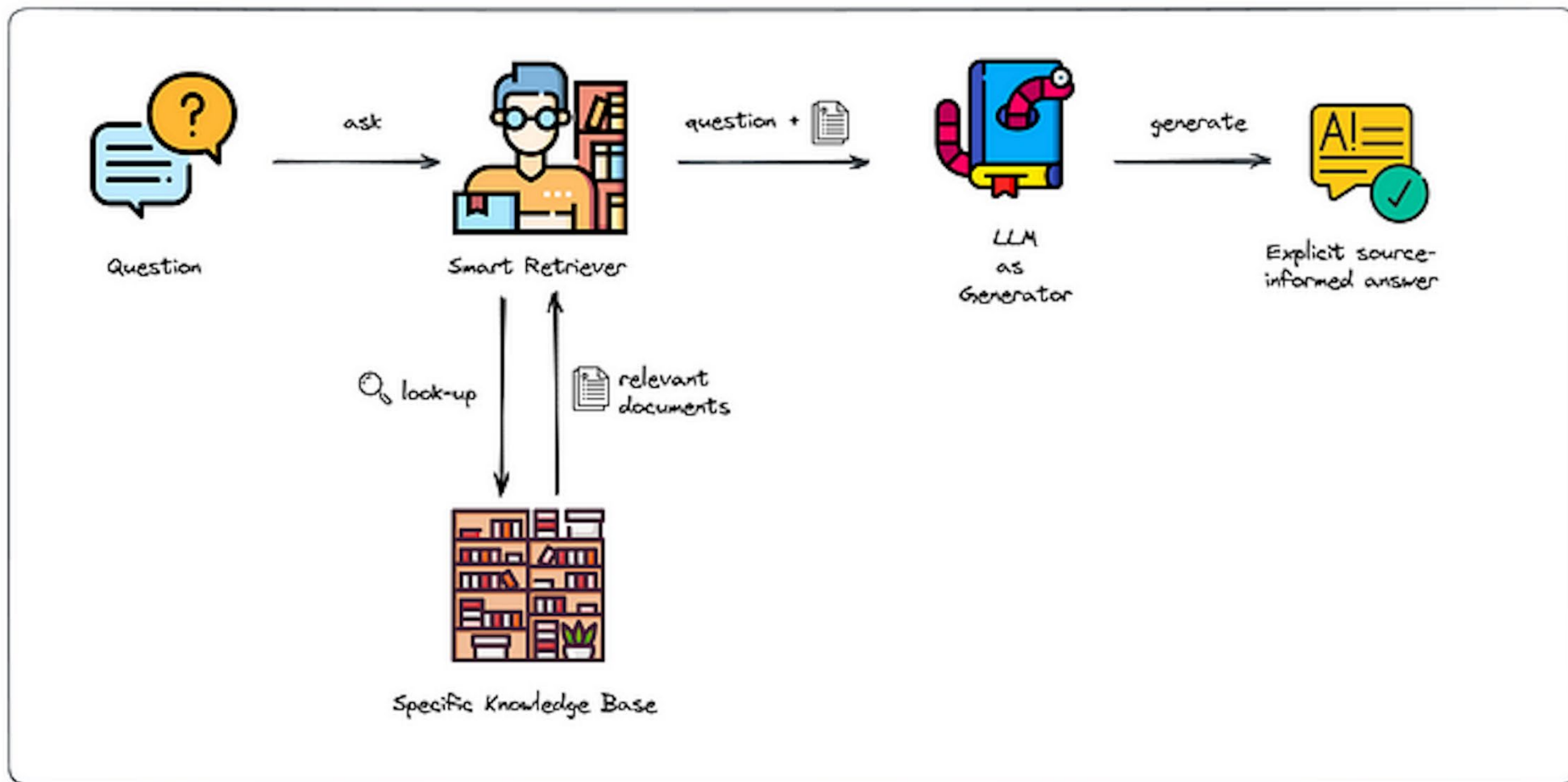
# 多样化的自然语言处理任务



# 教学科研对高效参数微调的需求



# ChatLLM/RAG等推理应用需求



# 基于MindSpore优势特性的NLP套件设计

NLP套件需求：

- 模型开发简便
- 高性能训练推理
- 简单高效的数据预处理
- 快速使用预训练模型



MindSpore优势特性：

- 函数式+面向对象融合编程
- 动静统一
- 数据处理引擎
- 自动并行

# MindSpore 2.x融合编程范式

1. 用类构建神经网络
2. 实例化Network对象
3. Network+Loss构造正向函数
4. 函数变换，获得梯度计算函数
5. 构造训练过程函数
6. 调用函数进行训练

```
net = Net()
loss_fn = nn.MSELoss()
optimizer = nn.Adam(net.trainable_params(), lr)

def forward_fn(inputs, targets):
    logits = net(inputs)
    loss = loss_fn(logits, targets)
    return loss

grad_fn = value_and_grad(forward_fn, None, optim.parameters)

def train_step(inputs, targets):
    loss, grads = grad_fn(inputs, targets)
    optimizer(grads)
    return loss

for i in range():
    for inputs, targets in dataset():
        loss = train_step(inputs, targets)
```



# MindSpore即时编译加速—— ms.jit

```
def train_step(inputs, targets):  
    loss, grads = grad_fn(inputs, targets)  
    optimizer(grads)  
    return loss
```



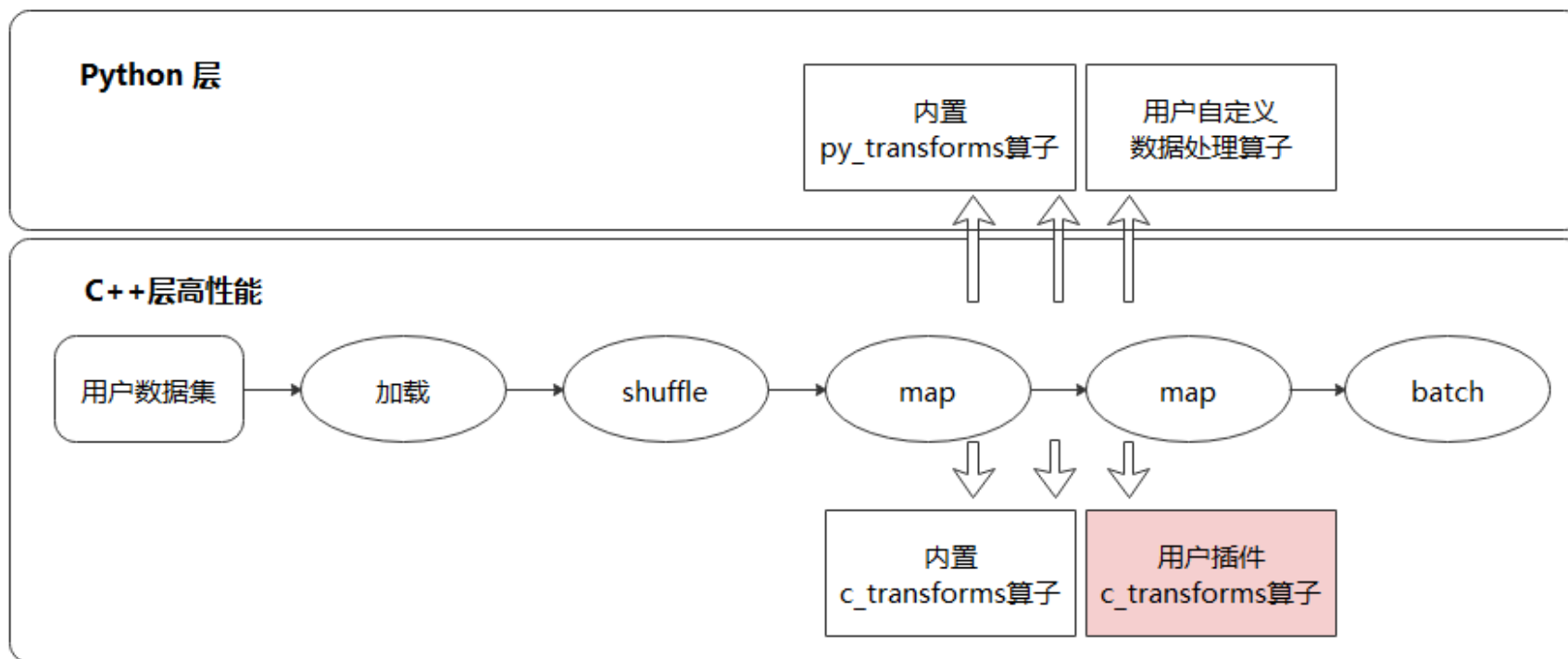
```
@ms.jit  
def train_step(inputs, targets):  
    loss, grads = grad_fn(inputs, targets)  
    optimizer(grads)  
    return loss
```

ms.jit修饰器:

- 一行代码切换动静态图
- 即时编译, 被修饰函数转为整图

# MindSpore Dataset引擎

高效数据处理Pipeline，让数据在Pipeline内流动，实现高效处理能力



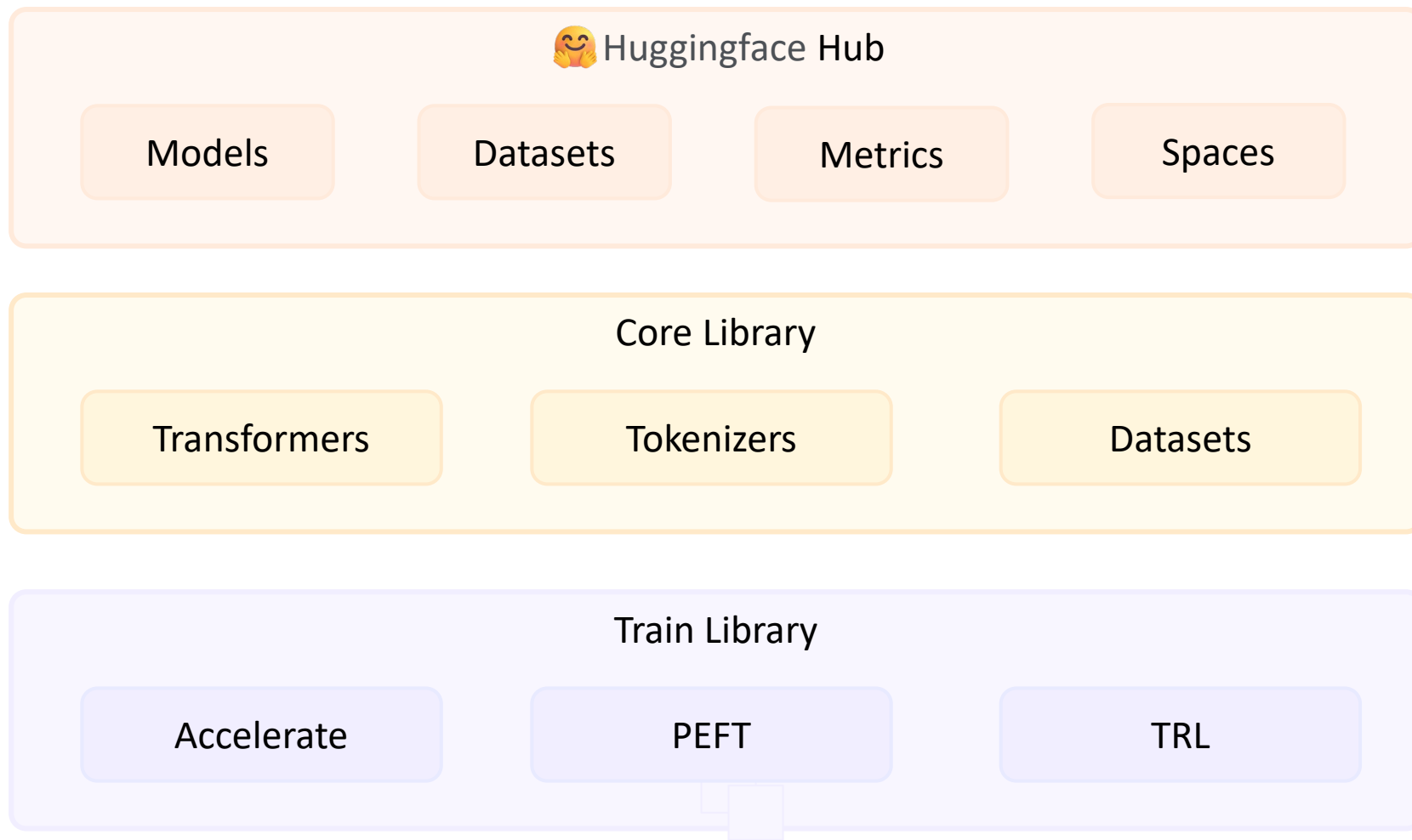
# 嫁接 🤗 生态——站在巨人的肩膀上

面对一个新框架的顾虑：

- 模型支不支持？
- 精度是否正确？
- 做模型修改是否方便？
- 迁移成本有多高？
- 已经有习惯使用的库怎么办？



# Huggingface ecosystem

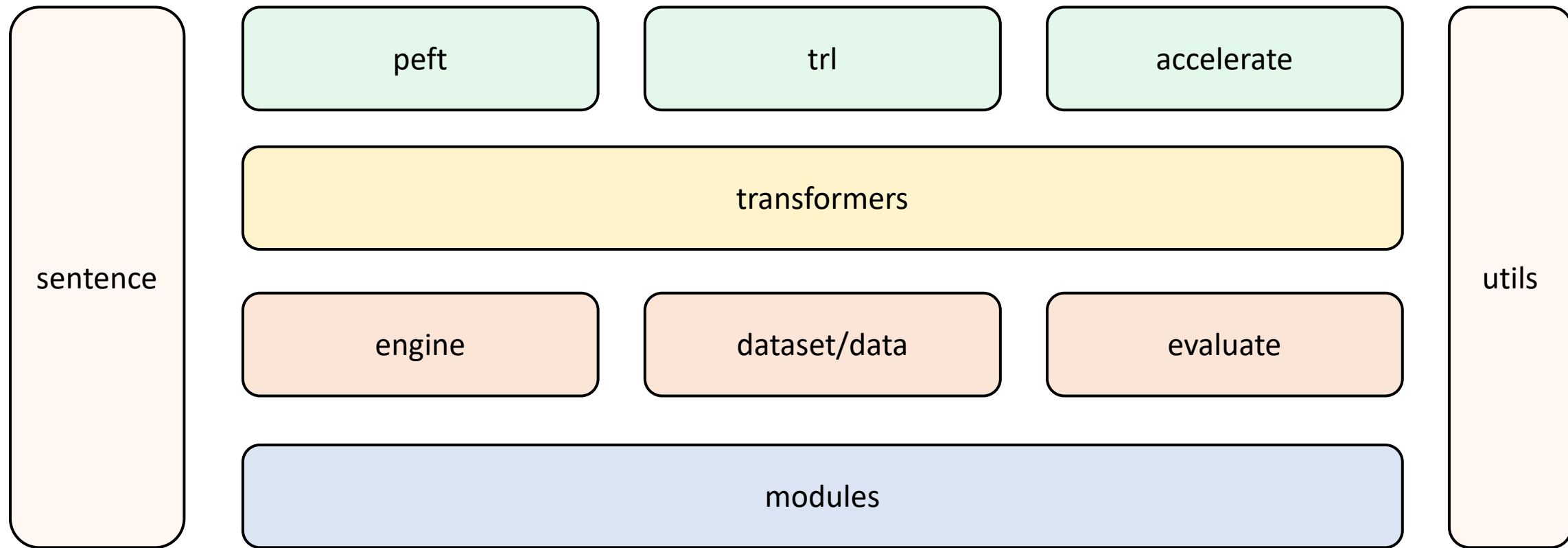


- 托管服务：模型、数据集、演示Demo

- 核心框架

- 训练框架：并行训练、高效参数微调、强化学习

# MindNLP架构



设计理念：顺应科研/开发者使用习惯，凸显MindSpore优势特性

# MindSpore数据引擎+ 🤗 Datasets

深度学习训练中的数据集和预处理痛点：

1. 数据格式不统一
2. 离线处理需要额外的实现
3. 完整数据集加载会消耗大量内存
4. 数据清洗需要大量工作量



**Datasets**

[huggingface.co/datasets](https://huggingface.co/datasets)



MindSpore

## Memory-mapping

😊 Datasets uses Arrow for its local caching system. It allows datasets to be backed by an on-disk cache, which is memory-mapped for fast lookup. This architecture allows for large datasets to be used on machines with relatively small device memory.

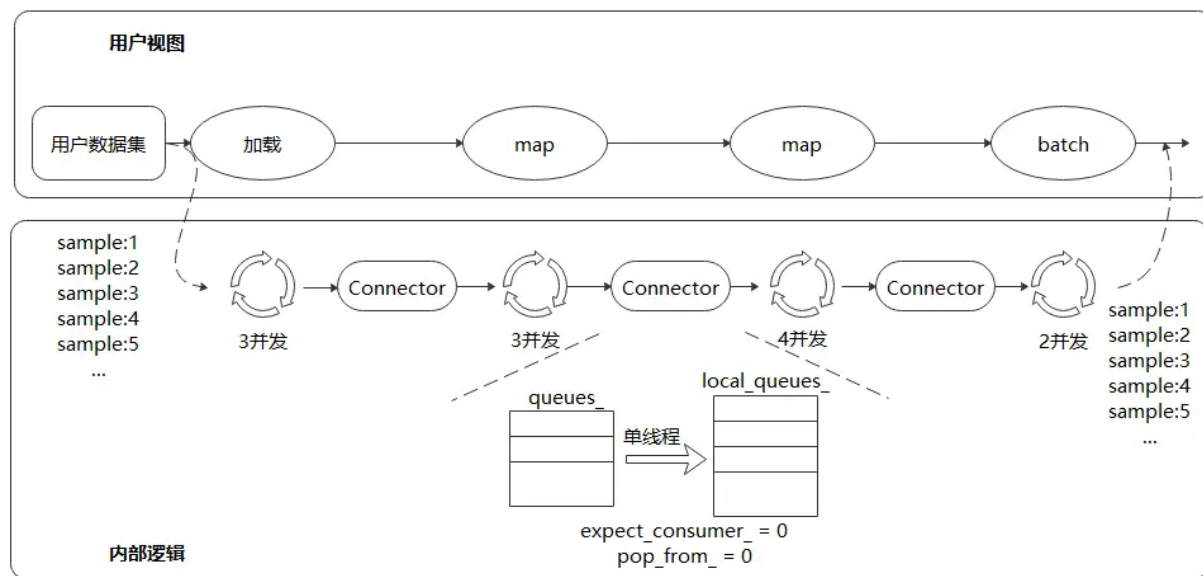
For example, loading the full English Wikipedia dataset only takes a few MB of RAM:

```
>>> import os; import psutil; import timeit
>>> from datasets import load_dataset

# Process.memory_info is expressed in bytes, so convert to megabytes
>>> mem_before = psutil.Process(os.getpid()).memory_info().rss / (1024 * 1024)
>>> wiki = load_dataset("wikipedia", "20220301.en", split="train")
>>> mem_after = psutil.Process(os.getpid()).memory_info().rss / (1024 * 1024)

>>> print(f"RAM memory used: {(mem_after - mem_before)} MB")
RAM memory used: 50 MB
```

This is possible because the Arrow data is actually memory-mapped from disk, and not loaded in memory. Memory-mapping allows access to data on disk, and leverages virtual memory capabilities for fast lookups.



## Pytorch:

- 数据集**全量**加载至内存 -> 全量遍历并预处理 -> 单条数据打包Batch -> 循环返回每个Batch

## MindNLP:

- 取**Batch size**条数据加载 -> Batch size条数据遍历并预处理 -> 返回一个Batch

# MindSpore数据引擎 + 🤗 Datasets

```
class TransferDataset():
    """TransferDataset for Huggingface Dataset."""
    def __init__(self, arrow_ds, column_names):
        self.ds = arrow_ds
        self.column_names = column_names

    def __getitem__(self, index):
        return tuple(self.ds[int(index)][name] for name in self.column_names)

    def __len__(self):
        return self.ds.dataset_size
```

```
class TransferIterableDataset():
    """TransferIterableDataset for Huggingface IterableDataset."""
    def __init__(self, arrow_ds, column_names):
        self.ds = arrow_ds
        self.column_names = column_names

    def __iter__(self):
        for data in self.ds:
            yield tuple(data[name] for name in self.column_names)
```

```
from datasets import load_dataset

imagenet = load_dataset("imagenet-1k", split="train")
print(imagenet[0])
```

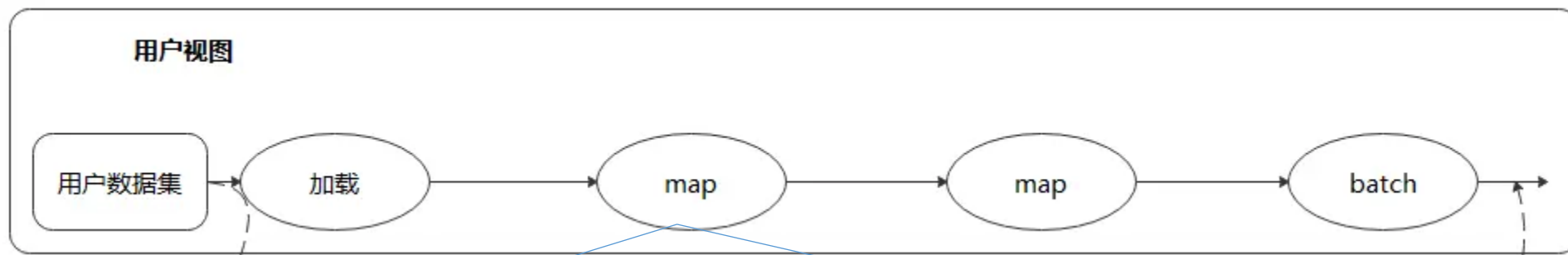


```
from mindnlp.dataset import load_dataset

imagenet = load_dataset("imagenet-1k", split="train")
print(imagenet[0])
```



# MindSpore数据引擎+ 🤗 Tokenizers



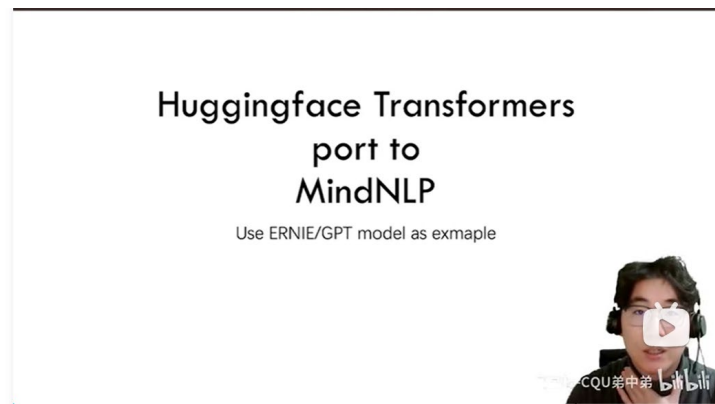
```
from mindnlp.dataset import BaseMapFuction

class MapFunc(BaseMapFuction):
    def __call__(self, sentence1, sentence2, label, idx):
        outputs = tokenizer(sentence1, sentence2, truncation=True, max_length=None)
        return outputs['input_ids'], outputs['attention_mask'], label
```

# 😊 全量模型支持

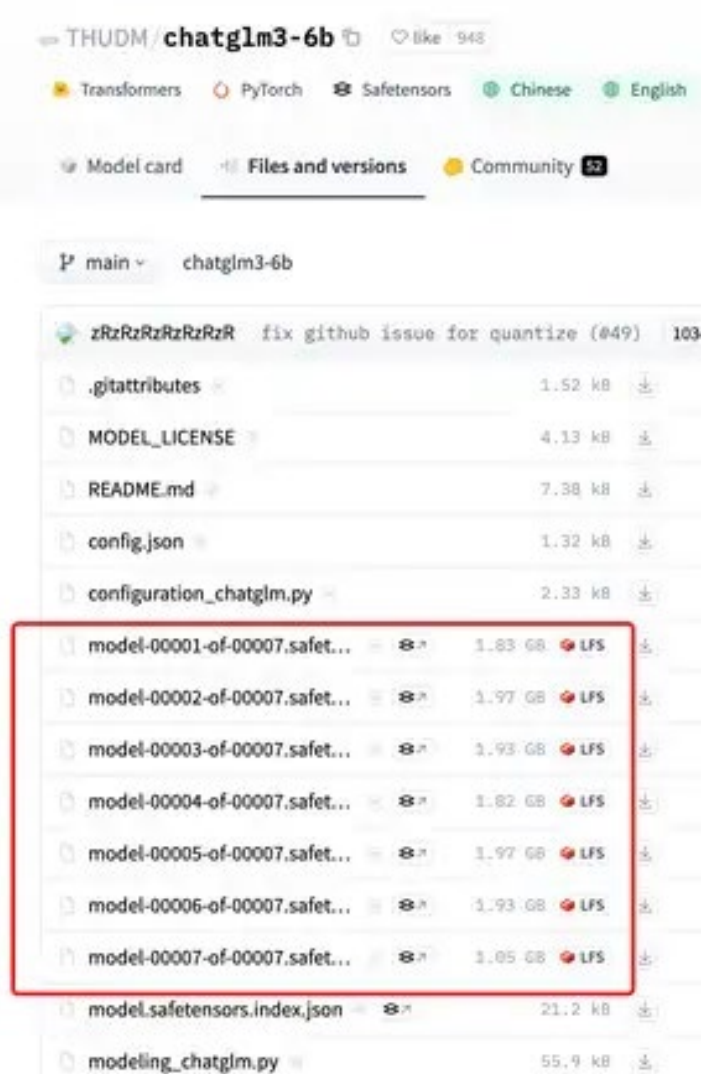
怎么做到和huggingface支持相同的模型？

- Transformers库同结构
- config、modeling、tokenization、  
processor迁移
- Huggingface全量UT测试
- Pytorch模型反序列化支持



| Name                                 |
|--------------------------------------|
| ..                                   |
| generation                           |
| integrations                         |
| models                               |
| pipelines                            |
| __init__.py                          |
| activations.py                       |
| audio_utils.py                       |
| backbone_utils.py                    |
| cache_utils.py                       |
| configuration_utils.py               |
| convert_slow_tokenizer.py            |
| feature_extraction_sequence_utils.py |
| feature_extraction_utils.py          |
| image_processing_utils.py            |
| image_transforms.py                  |
| image_utils.py                       |
| kernel_utils.py                      |
| modeling_attn_mask_utils.py          |
| modeling_outputs.py                  |
| modeling_utils.py                    |
| ms_utils.py                          |
| processing_utils.py                  |
| time_series_utils.py                 |
| tokenization_utils.py                |
| tokenization_utils_base.py           |
| tokenization_utils_fast.py           |

# Pytorch模型反序列化支持



```
import os
import platform

from mindnlp.transformers import ChatGLM3Tokenizer, ChatGLM3ForConditionalGeneration

tokenizer = ChatGLM3Tokenizer.from_pretrained("ZhipuAI/chatglm3-6b",
                                              mirror='modelscope',
                                              revision='master')

model = ChatGLM3ForConditionalGeneration.from_pretrained("ZhipuAI/chatglm3-6b",
                                                         mirror='modelscope',
                                                         revision='master')
```

MindNLP独有的  
反序列化模块

Pytorch模型  
无需转换!!!  
直接加载!!!



## Hugging Face 270+

| Model                         | Tokenizer slow | Tokenizer fast | PyTorch support | TensorFlow support | Flax Support |
|-------------------------------|----------------|----------------|-----------------|--------------------|--------------|
| ALBERT                        | ✓              | ✓              | ✓               | ✓                  | ✓            |
| ALIGN                         | ✗              | ✗              | ✓               | ✗                  | ✗            |
| AltCLIP                       | ✗              | ✗              | ✓               | ✗                  | ✗            |
| Audio Spectrogram Transformer | ✗              | ✗              | ✓               | ✗                  | ✗            |
| Autoformer                    | ✗              | ✗              | ✓               | ✗                  | ✗            |
| BART                          | ✓              | ✓              | ✓               | ✓                  | ✓            |
| BEiT                          | ✗              | ✗              | ✓               | ✗                  | ✓            |
| BERT                          | ✓              | ✓              | ✓               | ✓                  | ✓            |
| Bert Generation               | ✓              | ✗              | ✓               | ✗                  | ✗            |
| BigBird                       | ✓              | ✓              | ✓               | ✗                  | ✓            |
| BigBird-Pegasus               | ✗              | ✗              | ✓               | ✗                  | ✗            |
| BioGpt                        | ✓              | ✗              | ✓               | ✗                  | ✗            |
| BiT                           | ✗              | ✗              | ✓               | ✗                  | ✗            |
| Blenderbot                    | ✓              | ✓              | ✓               | ✓                  | ✓            |
| BlenderbotSmall               | ✓              | ✓              | ✓               | ✓                  | ✓            |
| BLIP                          | ✗              | ✗              | ✓               | ✓                  | ✗            |
| BLIP-2                        | ✗              | ✗              | ✓               | ✗                  | ✗            |
| BLOOM                         | ✗              | ✓              | ✓               | ✗                  | ✗            |
| BridgeTower                   | ✗              | ✗              | ✓               | ✗                  | ✗            |
| CamemBERT                     | ✓              | ✓              | ✓               | ✓                  | ✗            |
| CANINE                        | ✓              | ✗              | ✓               | ✗                  | ✗            |
| Chinese-CLIP                  | ✗              | ✗              | ✓               | ✗                  | ✗            |

## MindNLP 100+

| Model                         | Pynative support   | Graph Support |
|-------------------------------|--------------------|---------------|
| ALBERT                        | ✓                  | ✓             |
| ALIGN                         | ✓                  | ✗             |
| AltCLIP                       | ✓                  | ✗             |
| Audio Spectrogram Transformer | ✓                  | ✗             |
| Autoformer                    | ✓ (Inference only) | ✗             |
| BaiChuan                      | ✓                  | ✗             |
| Bark                          | ✓                  | ✗             |
| BART                          | ✓                  | ✗             |
| Barthez                       | ✓                  | ✗             |
| Bartpho                       | ✓                  | ✗             |
| BEiT                          | ✓                  | ✗             |
| BERT                          | ✓                  | ✗             |
| BERT Generation               | ✓                  | ✗             |
| BERT Japanese                 | ✓                  | ✗             |
| BERTweet                      | ✓                  | ✗             |
| BGE M3                        | ✓                  | ✗             |
| Bigbird                       | ✓                  | ✗             |
| Bigbird Pegasus               | ✓                  | ✗             |
| BioGPT                        | ✓                  | ✗             |
| Bit                           | ✓                  | ✗             |
| Blenderbot                    | ✓                  | ✗             |
| Blenderbot Small              | ✓                  | ✗             |
| BLIP                          | ✓                  | ✗             |
| BLIP2                         | ✓                  | ✗             |
| BLOOM                         | ✓                  | ✗             |

# 解决国内下载huggingface模型难

```
import os
import platform
from mindnlp.transformers import ChatGLM3Tokenizer, ChatGLM3ForConditionalGeneration

tokenizer = ChatGLM3Tokenizer.from_pretrained("ZhipuAI/chatglm3-6b",
                                              mirror='modelscope',
                                              revision='master')

model = ChatGLM3ForConditionalGeneration.from_pretrained("ZhipuAI/chatglm3-6b",
                                                         mirror='modelscope',
                                                         revision='master')
```

支持5大镜像站

- hf-mirror
- modelscope
- wisemodel
- gitee
- aifast

# PEFT算法支持

| Name            |
|-----------------|
| ..              |
| adalora         |
| adaption_prompt |
| ia3             |
| lokr            |
| lora            |
| prompt_tuning   |
| __init__.py     |
| tuners_utils.py |



支持常用高效参数微调方法，持续补齐中

# On going

- Accelerate: 接入MindFormers大模型预训练能力
- TRL: 对标hf的强化学习能力
- Sentence: 对标SentenceTransformers, 提供LLM向量编码能力
- Workflow: 自动化NLP训推任务执行

# MindNLP推理案例——ChatGLM

ChatGLM

Chatbot

你好

I

Submit

Clear History

Maximum length

2048

Top P

0.7

Temperature

0.95



# MindNLP推理案例——ChatGLM

```
import os
import platform
import signal
from mindnlp.transformers import AutoModelForSeq2SeqLM, AutoTokenizer

model = AutoModelForSeq2SeqLM.from_pretrained("THUDM/chatglm-6b").half()
model.set_train(False)
tokenizer = AutoTokenizer.from_pretrained("THUDM/chatglm-6b")

os_name = platform.system()
clear_command = 'cls' if os_name == 'Windows' else 'clear'
stop_stream = False

def build_prompt(history):
    prompt = "欢迎使用 ChatGLM-6B 模型，输入内容即可进行对话，clear 清空对话历史，stop 终止程序"
    for query, response in history:
        prompt += f"\n\n用户: {query}"
        prompt += f"\n\nChatGLM-6B: {response}"
    return prompt

def signal_handler(signal, frame):
    global stop_stream
    stop_stream = True

def main():
    history = []
    global stop_stream
    print("欢迎使用 ChatGLM-6B 模型，输入内容即可进行对话，clear 清空对话历史，stop 终止程序")
    while True:
        query = input("\n用户: ")
        if query.strip() == "stop":
            break
        if query.strip() == "clear":
            history = []
            os.system(clear_command)
            print("欢迎使用 ChatGLM-6B 模型，输入内容即可进行对话，clear 清空对话历史，stop 终止程序")
            continue
        count = 0
        for response, history in model.stream_chat(tokenizer, query, history=history):
            if stop_stream:
                stop_stream = False
                break
            else:
                count += 1
                if count % 8 == 0:
                    os.system(clear_command)
                    print(build_prompt(history), flush=True)
                    signal.signal(signal.SIGINT, signal_handler)
                    os.system(clear_command)
                    print(build_prompt(history), flush=True)

if __name__ == "__main__":
    main()
```

# MindNLP训练——Bloom微调

```
from mindspore import ops
```

```
from mindnlp.transformers import BloomTokenizerFast, BloomForCausalLM
from mindnlp.engine import TrainingArguments, Trainer
from mindnlp.dataset import load_dataset, BaseMapFuction
from mindnlp.amp import autocast
```

```
class ModifiedTrainer(Trainer):
    def compute_loss(self, model, inputs, return_outputs=False):
        return model(
            input_ids=inputs["input_ids"],
            attention_mask=ops.ones_like(inputs["input_ids"]).bool(),
            labels=inputs["input_ids"],
        ).loss
```

```
model_name = "bloom-560m"
model = BloomForCausalLM.from_pretrained(f"bigscience/{model_name}")
tokenizer = BloomTokenizerFast.from_pretrained(f"bigscience/{model_name}", add_prefix_space=True)
```

```
dataset = load_dataset('tatsu-lab/alpaca')
print(dataset.get_col_names())
```

```
class ModifiedMapFunction(BaseMapFuction):
    def __call__(self, text):
        tokenized = tokenizer(text, max_length=512, padding="max_length", truncation=True)
        return tokenized['input_ids']
```

- 自定义Trainer
- 加载模型和Tokenizer
- 加载数据集
- 定义map函数

# MindNLP训练——Bloom微调

```
training_args = TrainingArguments(  
    "output",  
    fp16=False,  
    gradient_accumulation_steps=1,  
    per_device_train_batch_size=2,  
    learning_rate=2e-5,  
    num_train_epochs=2,  
    logging_steps=100,  
    save_strategy='epoch'  
)
```

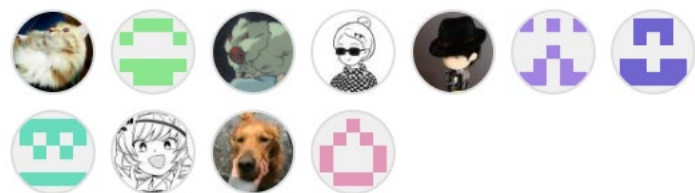
```
trainer = ModifiedTrainer(  
    model=model,  
    train_dataset=dataset,  
    args=training_args,  
    map_fn=ModifiedMapFunction('text', 'input_ids'),  
)
```

```
trainer.train()
```

- 配置训练参数
- 实例化Trainer
- 开始训练

# MindNLP社区贡献

Contributors 38



+ 27 contributors

代码贡献者74名，其中36名  
高校同学，包括香港科技大学、浙江大学、华东师范大学、重庆大学、武汉理工大学、等高校共同参与贡献

依托MindSpore社区实习，  
快速支持大量预训练模型

【开源实习】预训练模型nezha intern intern-task-assigned

#I6GFNL 杨宇澄 14

【开源实习】预训练模型mobilebert intern intern-task-assigned

#I6GFMM 杨宇澄 6

【开源实习】预训练模型megatron\_gpt2 intern intern-task-assigned

#I6GFME 杨宇澄 6

【开源实习】预训练模型megatron\_bert intern intern-task-assigned

#I6GFMS 杨宇澄 10

【开源实习】预训练模型longformer intern intern-task-assigned

#I6GFKQ 杨宇澄 7

【开源实习】预训练模型gpt\_neo intern intern-task-assigned

#I6GFJS 杨宇澄 6

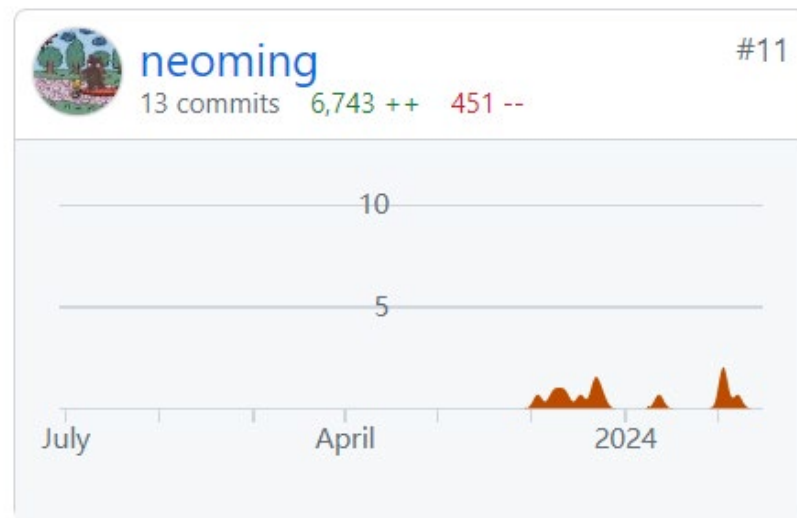
# 高校同学的社区贡献



重庆大学李佳明同学：

- T5预训练模型
- PreTrainedModel文本生成功能 (model.generate)

代码量2.3w+， PR数39



华东师范大学李一鸣同学：

- 1天内完成GPT bigcode代码迁移
- MindSpore自动并行预训练
- 完成代码迁移+精度验证+单卡微调+多卡训练全流程任务

# MindSpore社区实习

[M]<sup>s</sup>昇思  
MindSpore

## 开源实习!

MindSpore

## 2024 任务开启!

MindSpore

你有新的开源实习任务待领取

2024 →

昇思MindSpore社区联合多个开源社区为高校学生推出线上实习的机会——开源实习，让你不用出校门就可以拥有在业界顶尖AI开源项目实践的机会。



### 02 面向对象

- ① 年满18周岁在校大学生，无专业年级限制
- ② 平均每周能投入10小时以上的时间

### 03 线上实习

没有地域限制!

获得对应任务积分!

即可领取实习薪资及实习证明!



### 04 新任务看这里

- 1 开发者活动类**

实习期内按任务要求组织开发者活动，促进更多开发者对人工智能的了解和对昇思MindSpore的掌握!
- 2 开源贡献类**

实习期内按要求完成任务并合入PR。

  - 套件开发
  - 模型迁移
  - 文档完善
  - API整改

### 05 扫码申请

昇思小助手微信号: mindspore0328



实习申请



任务领取

# 欢迎使用扫码使用



## MindNLP

[docs](#)[latest](#)[license](#)[Apache-2.0](#)[PRs](#)[welcome](#)[issues](#)[13 open](#)[CI Pipe](#)[passing](#)

[Introduction](#) | [Quick Links](#) | [Installation](#) | [Get Started](#) | [Tutorials](#) | [Notes](#)

## News 📢

- 🔥 Latest Features
  - 📄 Support PreTrained Models, including [BERT](#), [Roberta](#), [GPT2](#) and [T5](#). You can use them by following code snippet:

```
from mindnlp.models import BertModel

model = BertModel.from_pretrained('bert-base-cased')
```

## Introduction

MindNLP is an open source NLP library based on MindSpore. It supports a platform for solving natural language processing tasks, containing many common approaches in NLP. It can help researchers and developers to construct and train models more conveniently and rapidly.

The master branch works with **MindSpore master**.

Q&A