浙江大学
ZHEJIANG UNIVERSITY

完成一篇论文的科研历程与经验

报告人：彭思达

GitHub Repo: https://github.com/pengsida/learning_research

Agenda

一个research project包含哪些流程，怎么做一篇论文

a. 如何想Idea
b. 如何做实验
c. 如何写论文

一个research project包含哪些流程

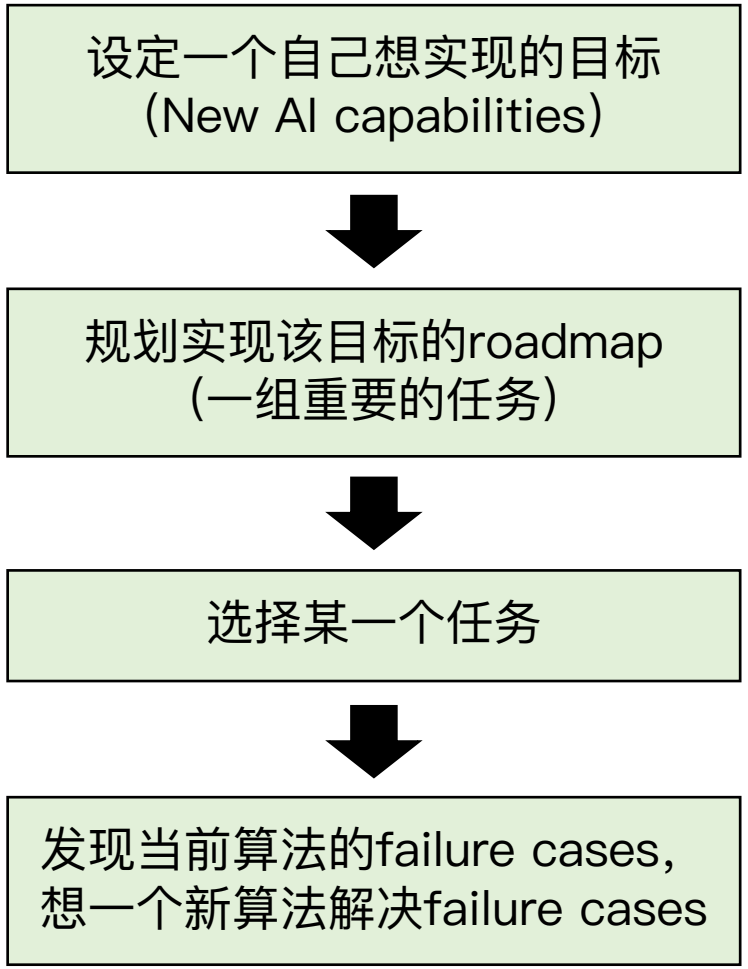
1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 根据论文的截止日期设置自己论文重要的几个截止时间
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

1. 如何想Idea

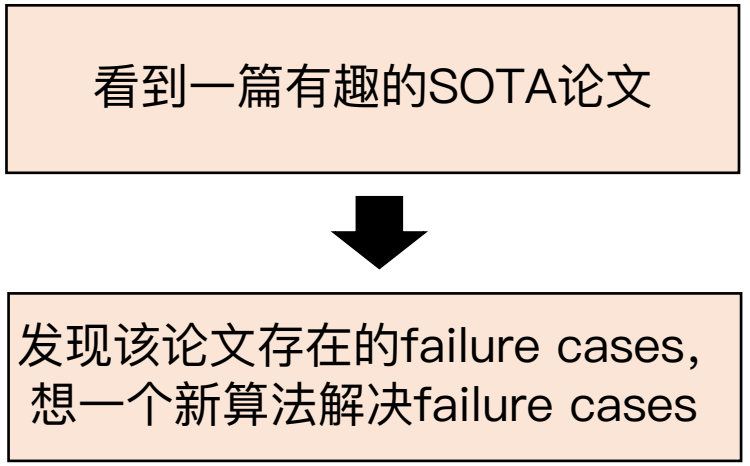
1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

我把这样的研究风格称为Goal-driven research，与之相对的还有Idea-driven research。

Goal-driven research vs. Idea-driven research



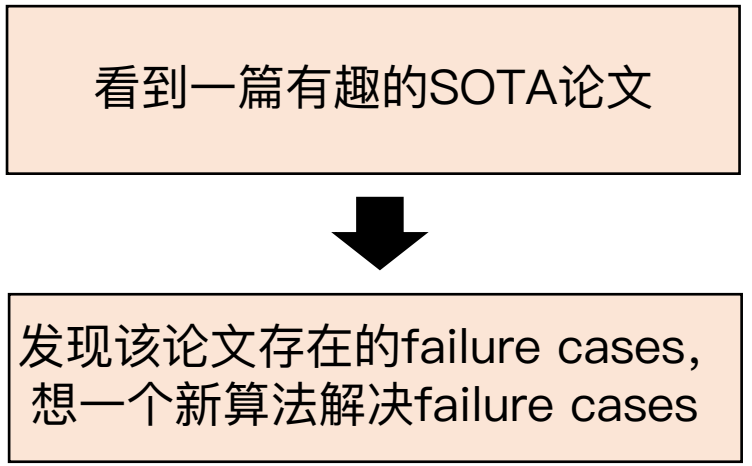
Goal-driven research



Idea-driven research

Idea-driven research的优缺点

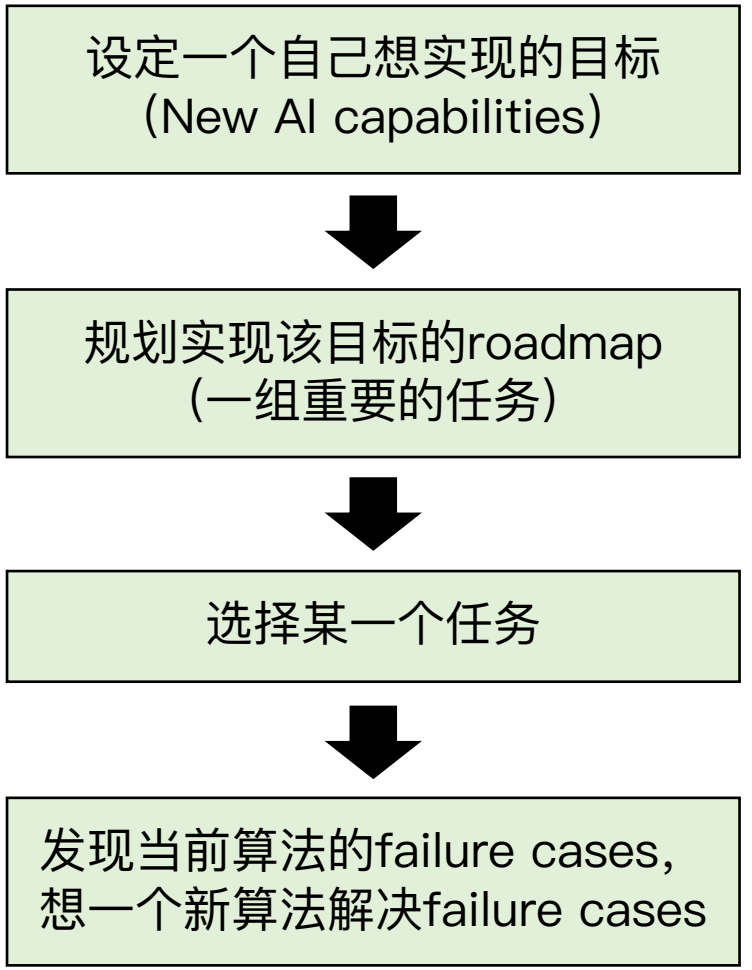
- 优点：容易上手。这应该是大部分新手的科研方式。
- 缺点：
  1. 因为是follow别人，所以除非效果好一大截，不然论文影响力有限。
  2. 可能被这篇论文带偏。该论文路线可能是错的。
  3. 容易和别人撞idea，因为其他人也可能在改进这篇论文。
  4. 因为已经有一些解法了，所以解法上的创新空间有限。需要很聪明、有很深刻的见解，才能为这个task提出突破性的new technique。



Idea-driven research

Goal-driven research的优缺点

- 优点：
  1. 容易做出突破性的创新，因为我们不再是follow-up，而是引领者。
  2. 自己的科研更有motivation，有大的picture，容易吸引他人。
  3. 科研更具有连续性，成体系。
- 缺点：
  1. 规划一个好的roadmap是一个很难的事情。



Goal-driven research

1.1 如何规划Project要解决的任务

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的解法

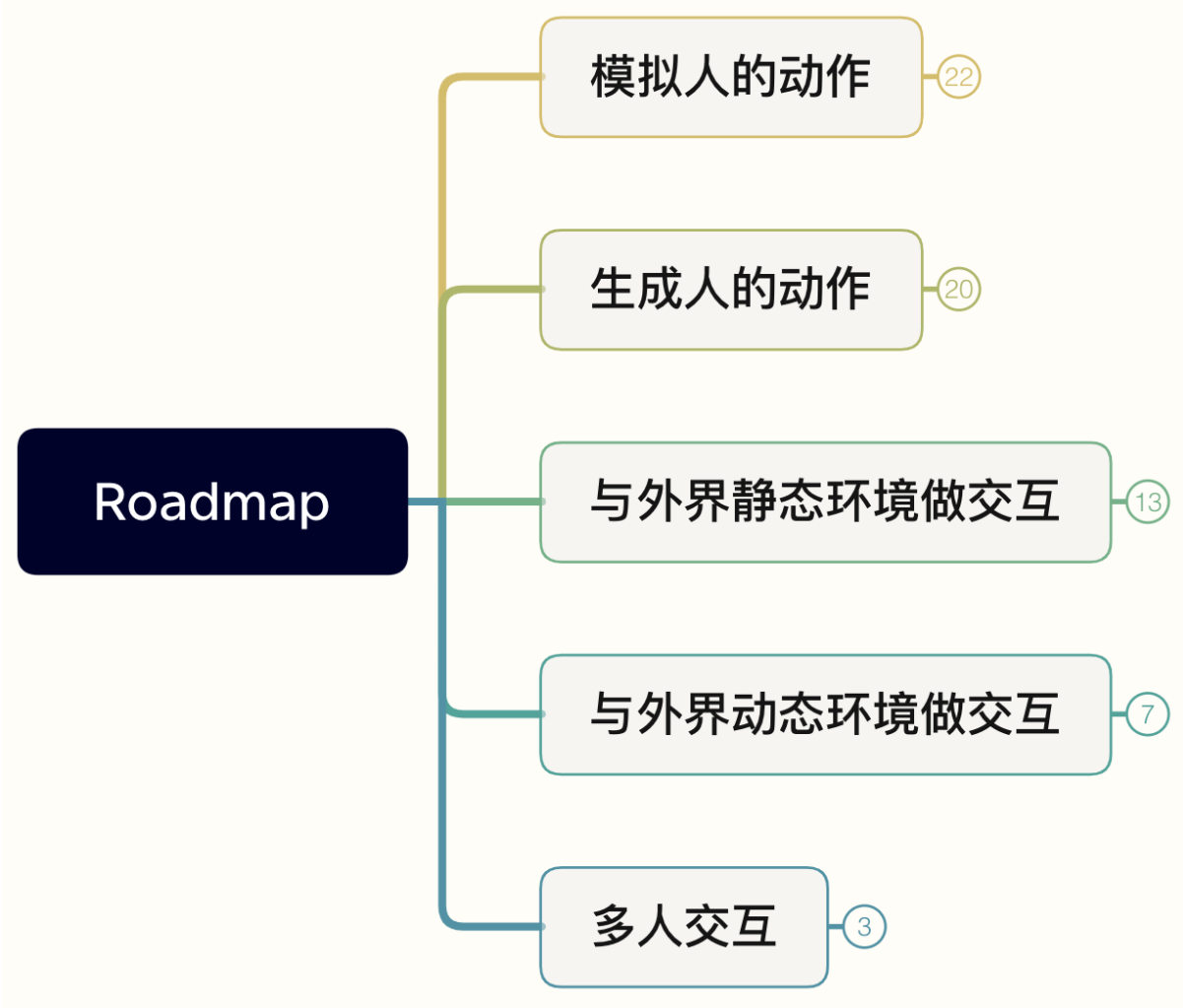
1.1 如何规划Project要解决的任务

- 首先，设定一个自己的长期科研目标。

  比如，让ChatGPT拥有身躯，实现一个有自己身躯的AI agent。

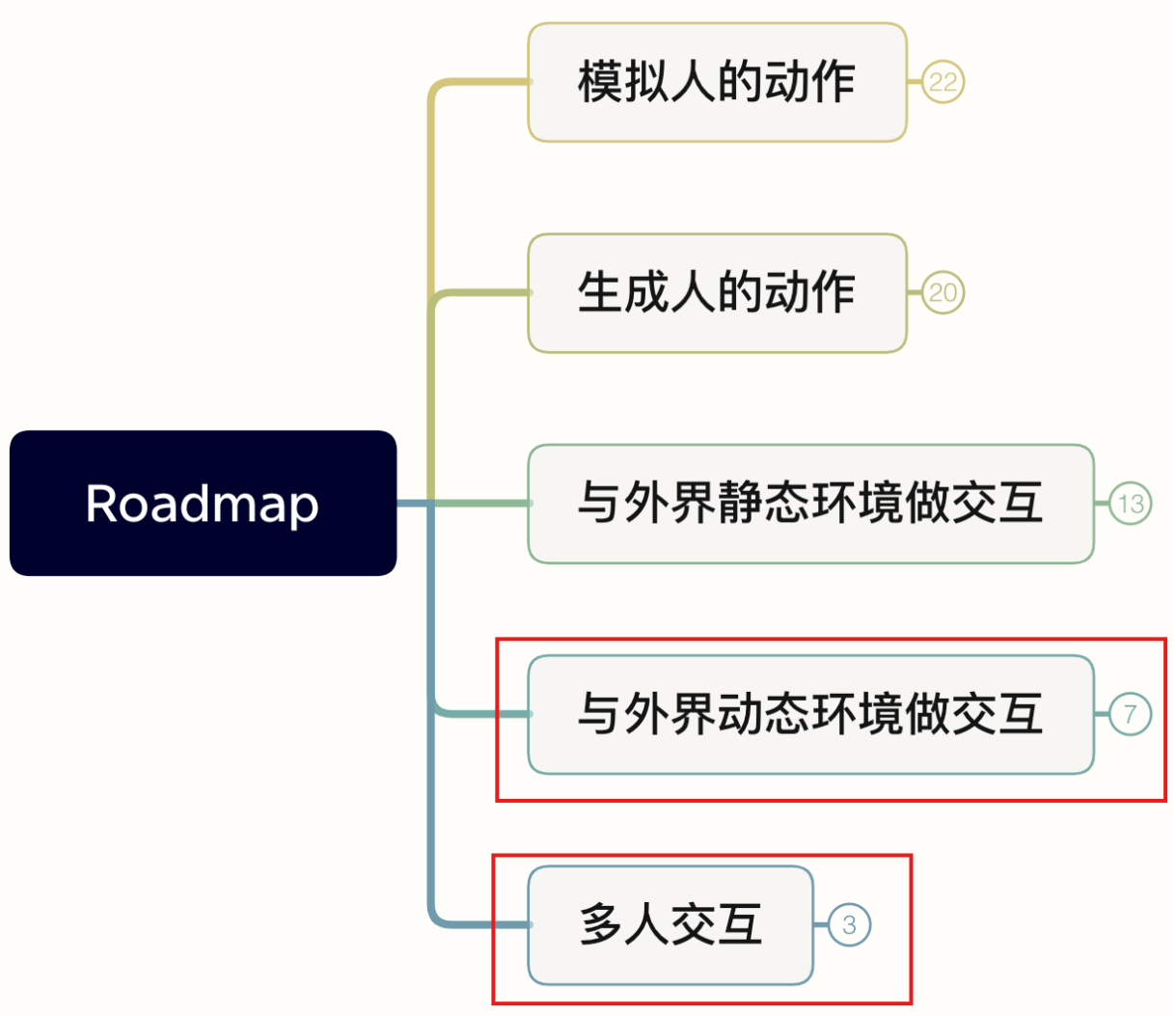
1.1 如何规划Project要解决的任务

- 然后，规划实现该目标的roadmap，制定一组重要的任务。



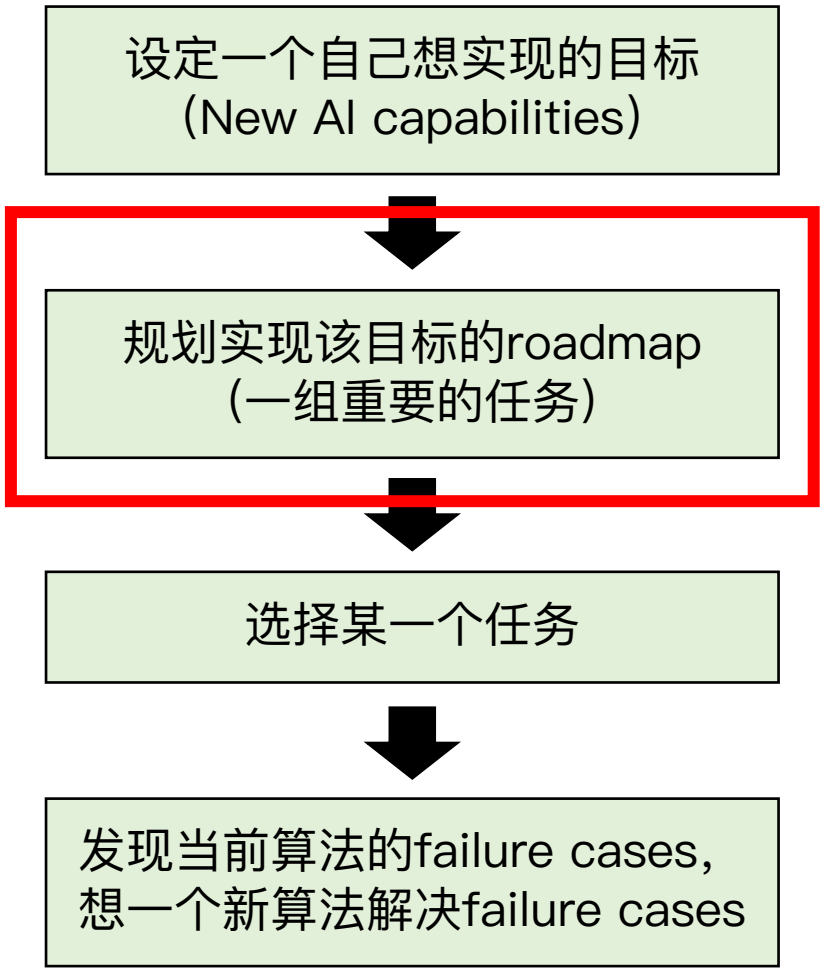
1.1 如何规划Project要解决的任务

- 最后，根据研究空间与当前学术界的技术发展情况，选择合适的任务。



1.1.2 选择任务的关键：规划roadmap

规划Roadmap是最重要的一步，也是最难的一步。



Goal-driven research

1.1.1 如何制定roadmap：文献调研

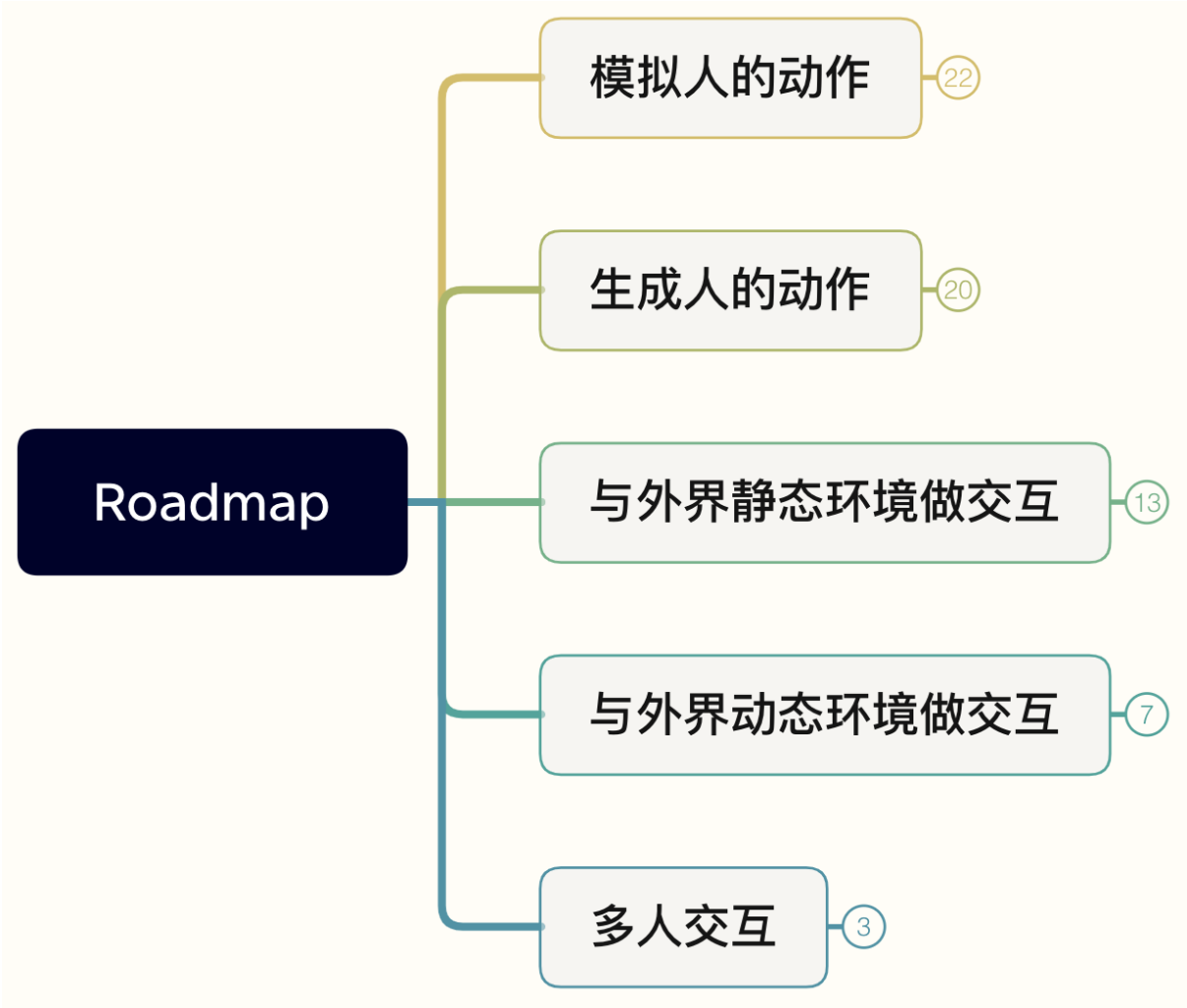
通过literature tree帮助自己：

1. 列出自己科研方向的大多数论文。

1.1.1 如何制定roadmap： 文献调研

通过literature tree帮助自己：

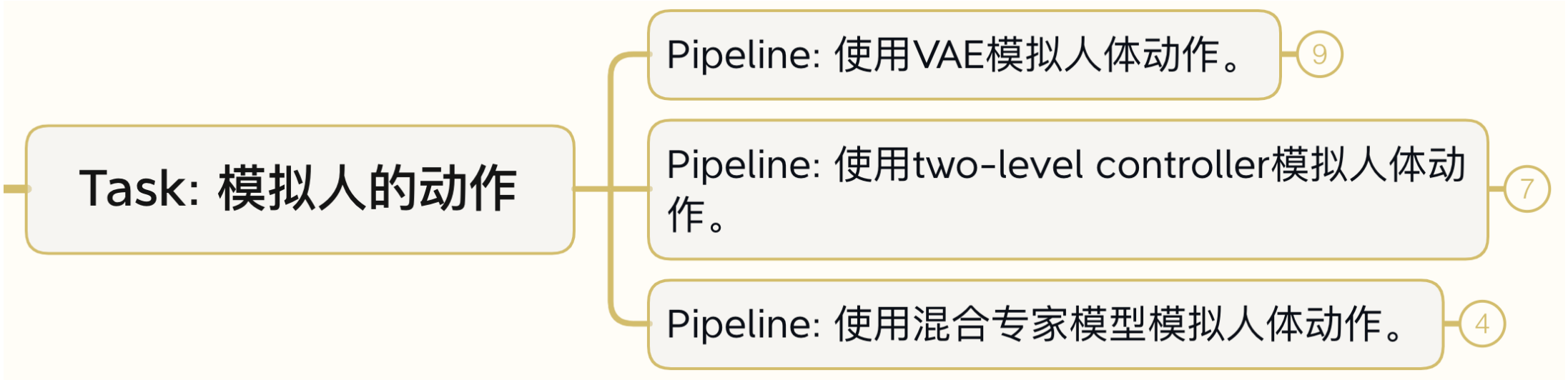
2. 通过阅读论文，梳理出当前方向已有的milestone tasks，以此归类论文。



1.1.1 如何制定roadmap： 文献调研

通过literature tree帮助自己：

3. 对于每一个milestone task，梳理出有代表性的pipelines，并以此归类论文。

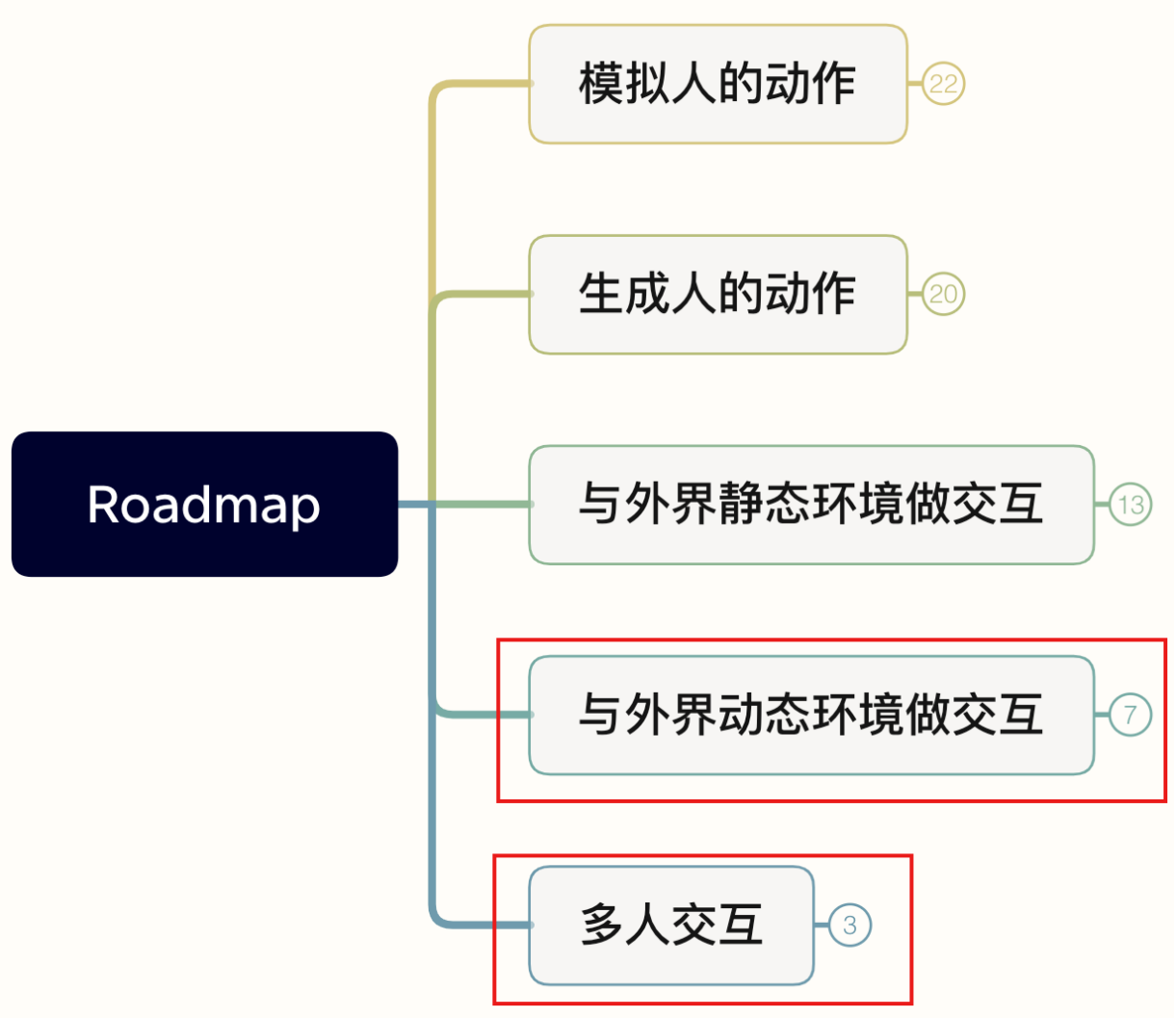


1.1.1 如何制定roadmap： 文献调研

通过文献树（literature tree）判断：

- 哪些任务是重要的?
- 哪些任务的研究空间较大?

从而帮助自己制定Roadmap。



1.2 如何发现任务中需要解决的failure cases

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

1.2 如何发现任务中需要解决的failure cases

- 现有的问题：经常有一些同学觉得某某任务没啥好做的了，被某篇论文做完了。

- 出现这个问题的原因：论文总是会把最好的结果放上去，营造出效果很好的感觉。

- 如果我们换到其他数据上，其实很可能效果不好。

1.2 如何发现任务中需要解决的failure cases

- 探索算法的上限，尝试更具有挑战性的cases（通常是数据）。
- 在**新的task setting**或者**新的数据**上容易发现新的failure cases。

1.2 如何发现任务中需要解决的failure cases

- 探索算法的上限，尝试更具有挑战性的cases（通常是数据）。
- 在新的task setting或者新的数据上容易发现新的failure cases。

- 注意：在新数据上探索方法的可能性，**让大家看到新的实验结论，**这是很大的贡献。

1.3 如何构思解决failure cases的方法

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法

1.3 如何构思解决failure cases的方法

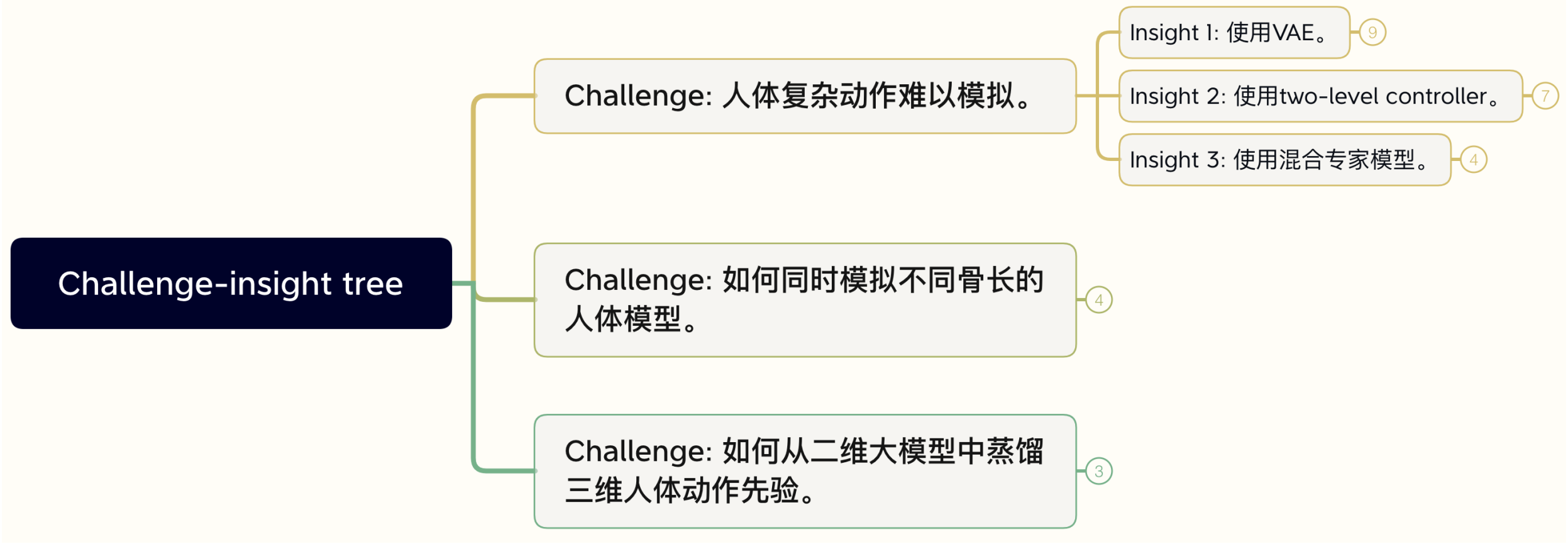
怎么去提出有效的解法：**积累自己的武器库。**（背后的道理：很多任务中存在相似的technical challenge，而解决这些technical challenge的技术是通用的）

1.3 如何构思解决failure cases的方法

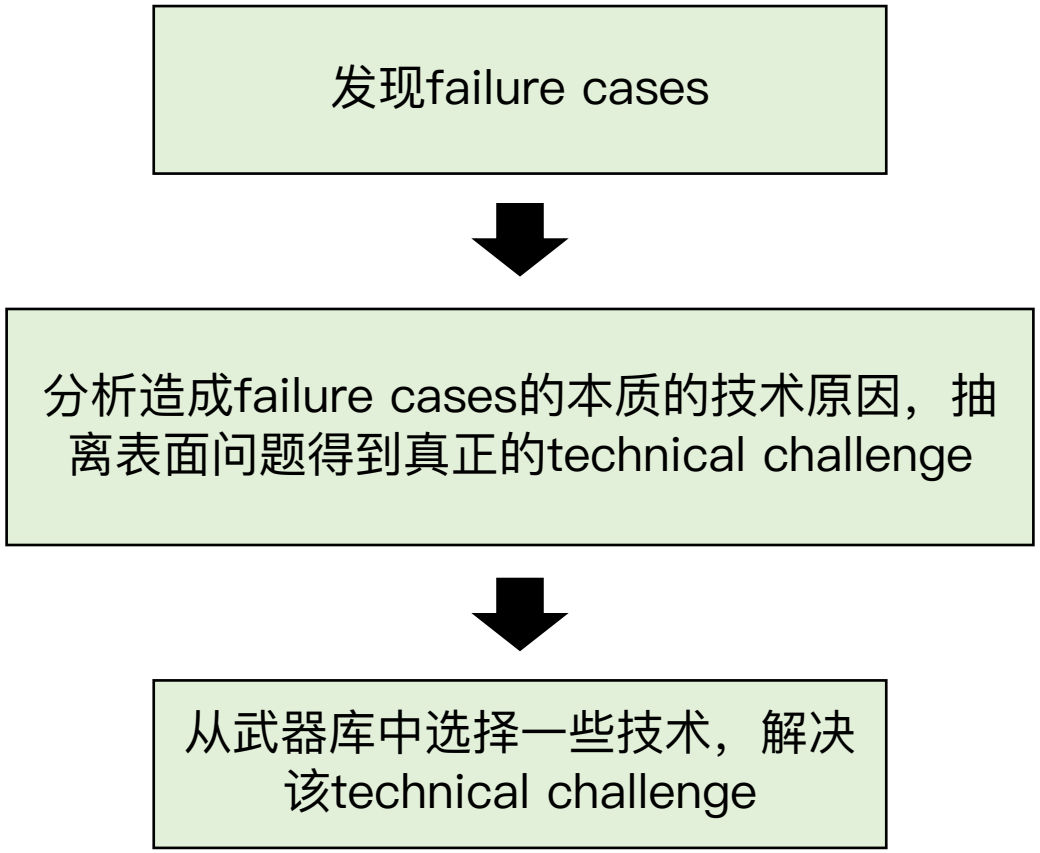
怎么去提出有效的解法：**积累自己的武器库。**（背后的道理：很多任务中存在相似的technical challenge，而解决这些technical challenge的技术是通用的）

怎么积累自己的武器库：**我的做法是构建challenge–insight tree。**帮助自己知道这个研究方向存在哪些technical challenges，而有哪些techniques/insights在尝试解决这些technical challenges。

举个challenge–insight tree的例子（简化版）



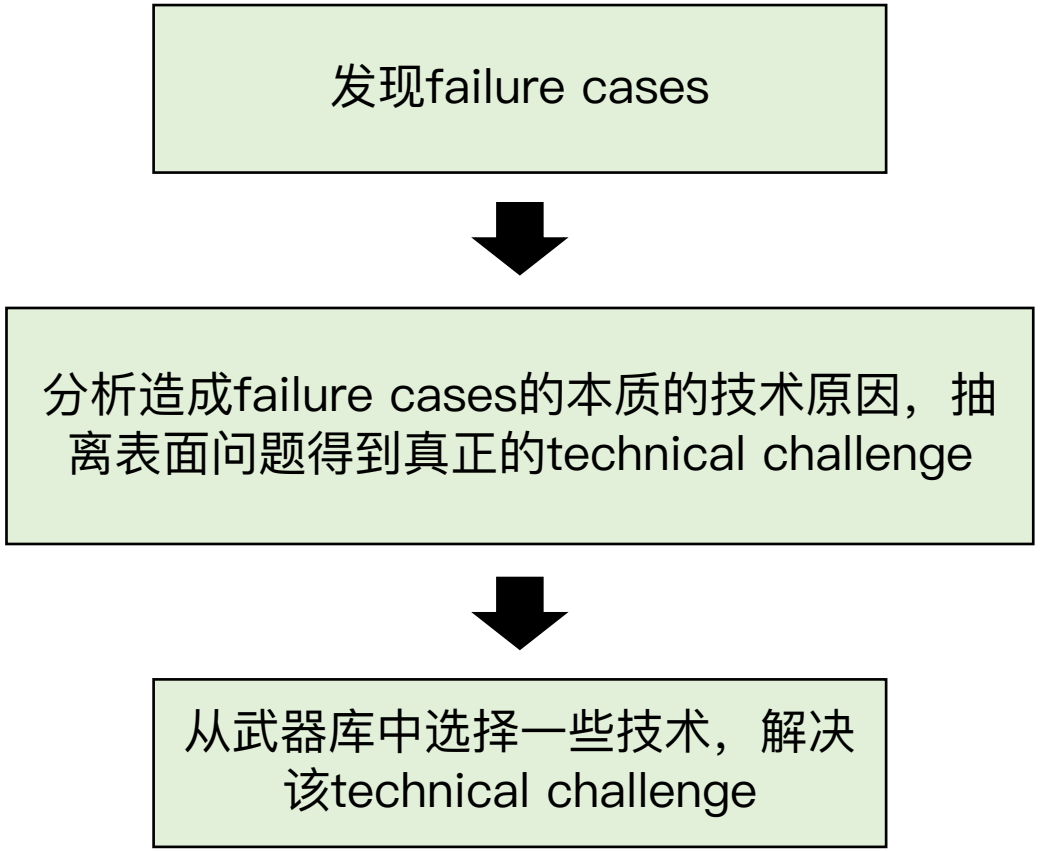
1.3 如何构思解决failure cases的方法



1.3.1 解决重要failure cases的方法一定是novel的

- 证明：
  1. 如果简单的技术组合即可解决该failure cases，那么该failure cases并不重要。
  2. 如果failure cases很新颖，则无法被简单技术组合所解决，则会促使我们提出新的技术。

Failure cases要尽量general。



2. 如何做实验

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work

2.1 如何设计简单实验快速验证解法的正确性

- 什么是简单实验：只存在core technical challenge的实验

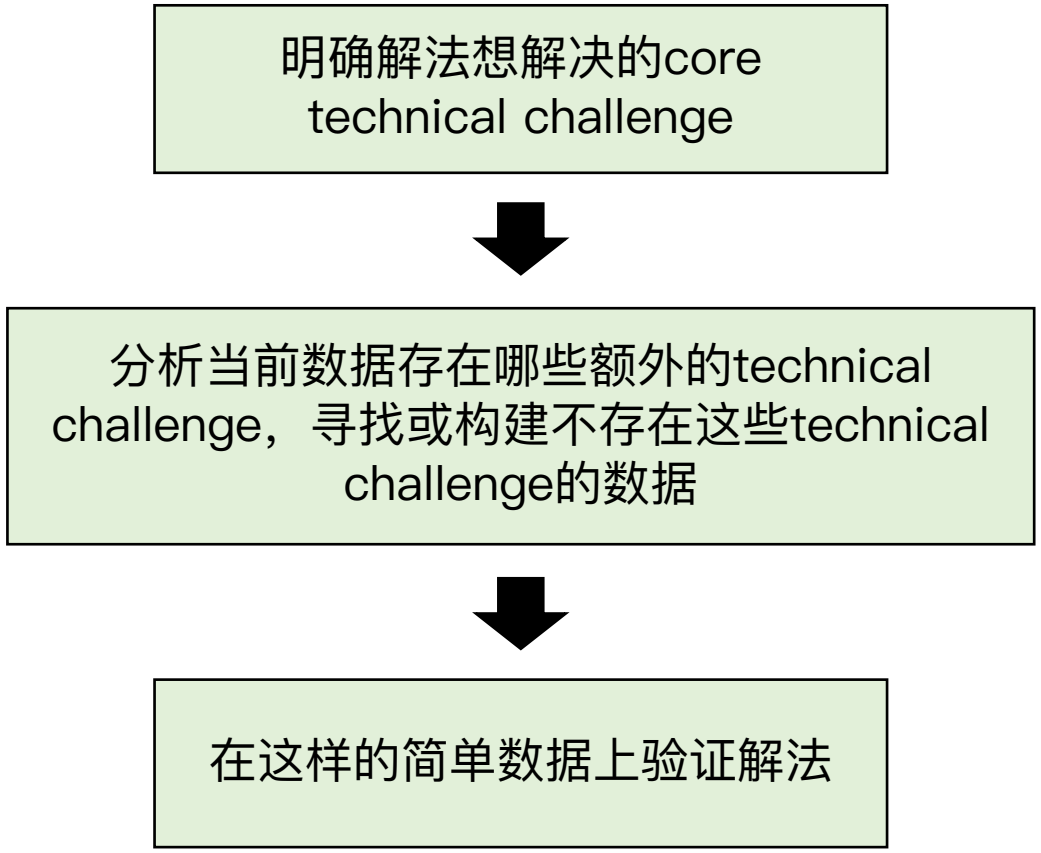
通过简单实验验证解法的好处：

- 实验成本低
- 不存在其他technical challenges的干扰，可以更好地确定当前的解法是否work

2.1 如何设计简单实验快速验证解法的正确性

- 什么是简单实验：只存在core technical challenge的实验

- 如何设计简单实验

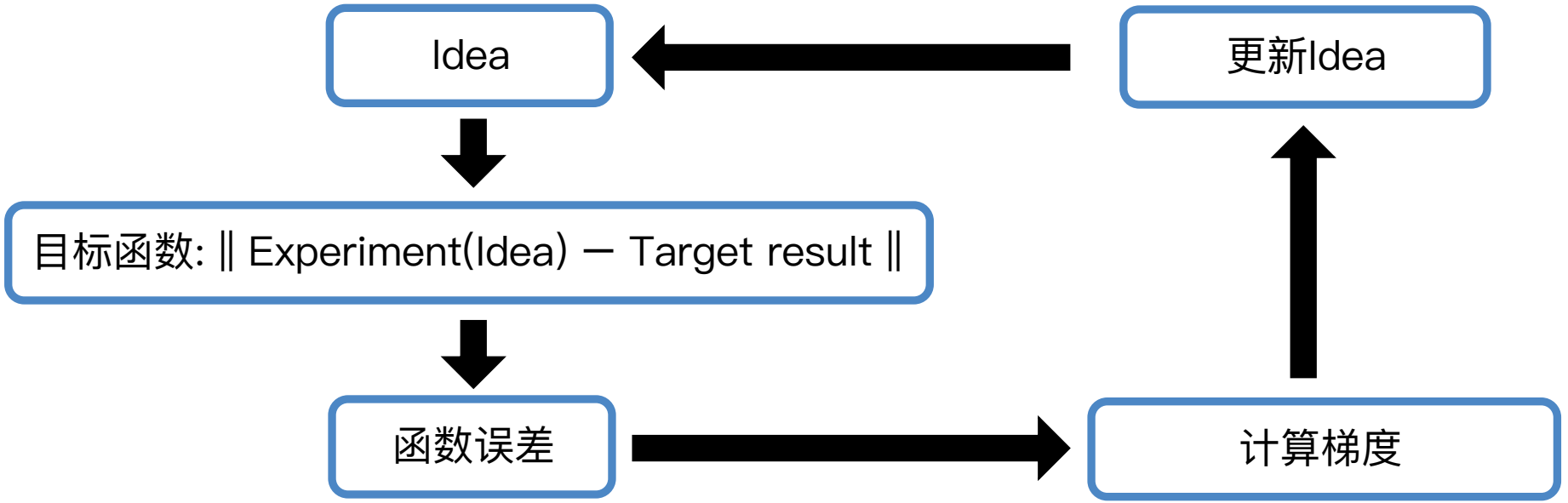


2.2 在简单数据上，改进解法、畅想更多解法

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work

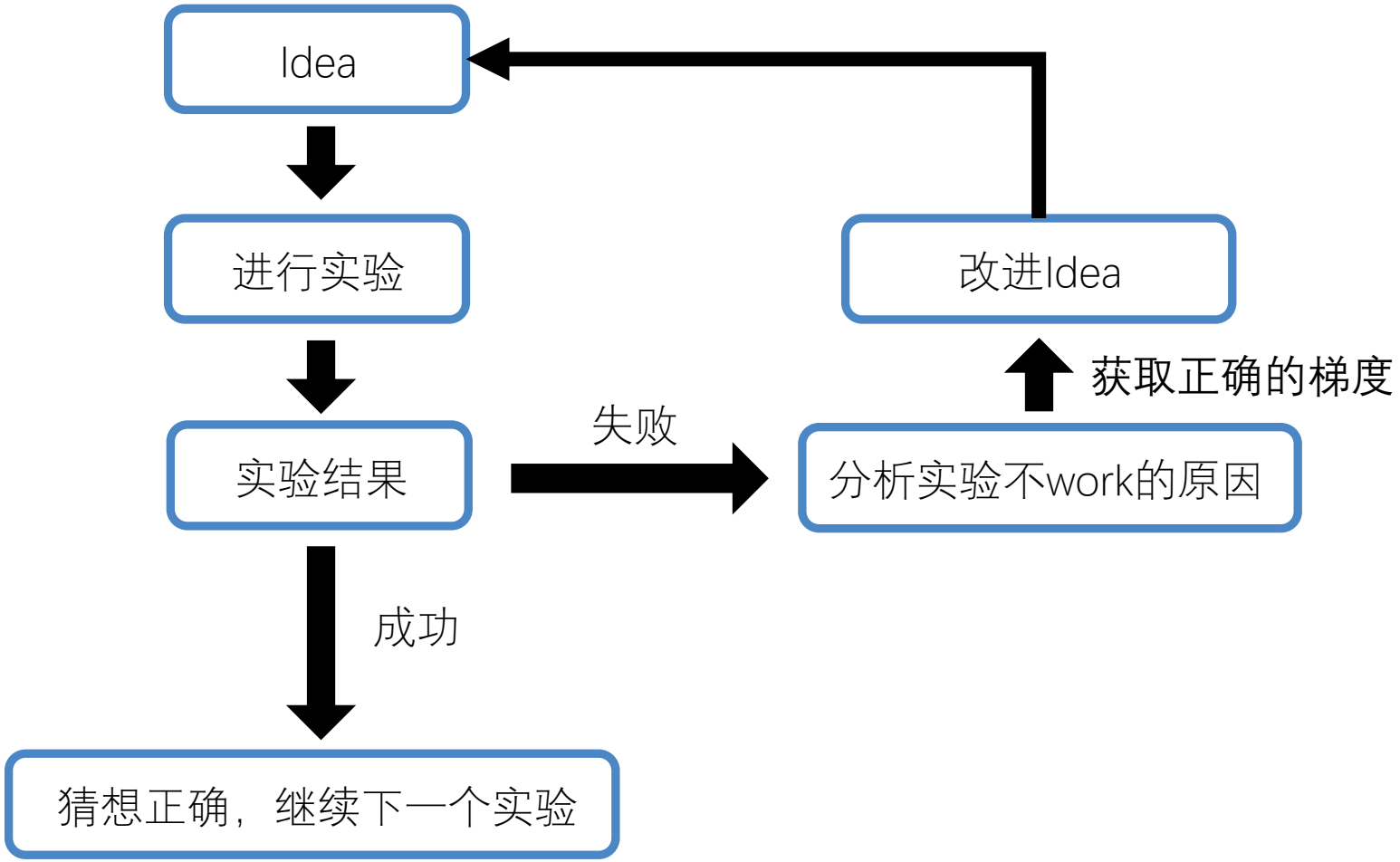
2.2 改进解法、畅想更多解法

- 把做探索实验的过程视为基于SGD优化Idea的过程



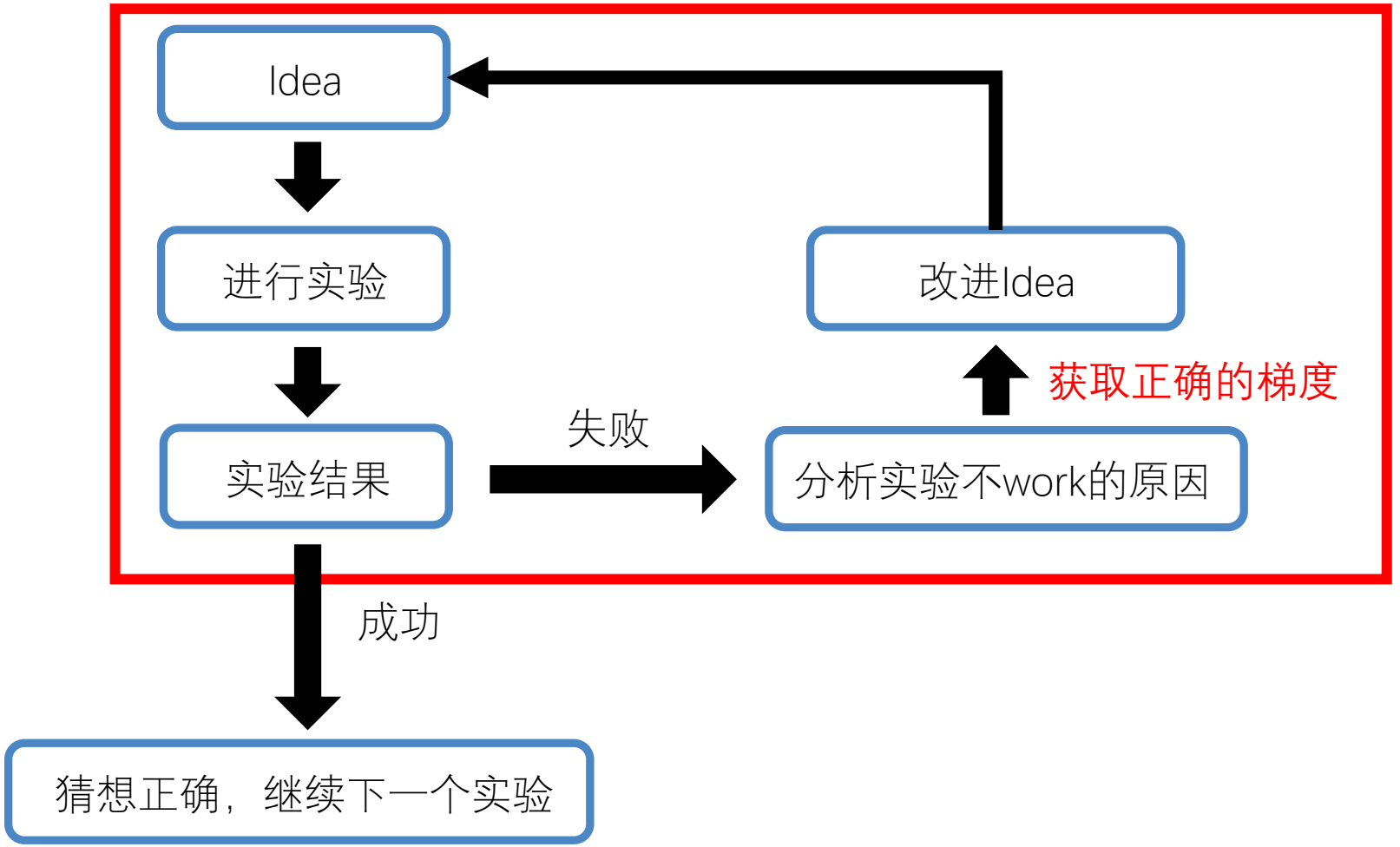
2.2 改进解法、畅想更多解法

- 通过实验的反馈优化Idea



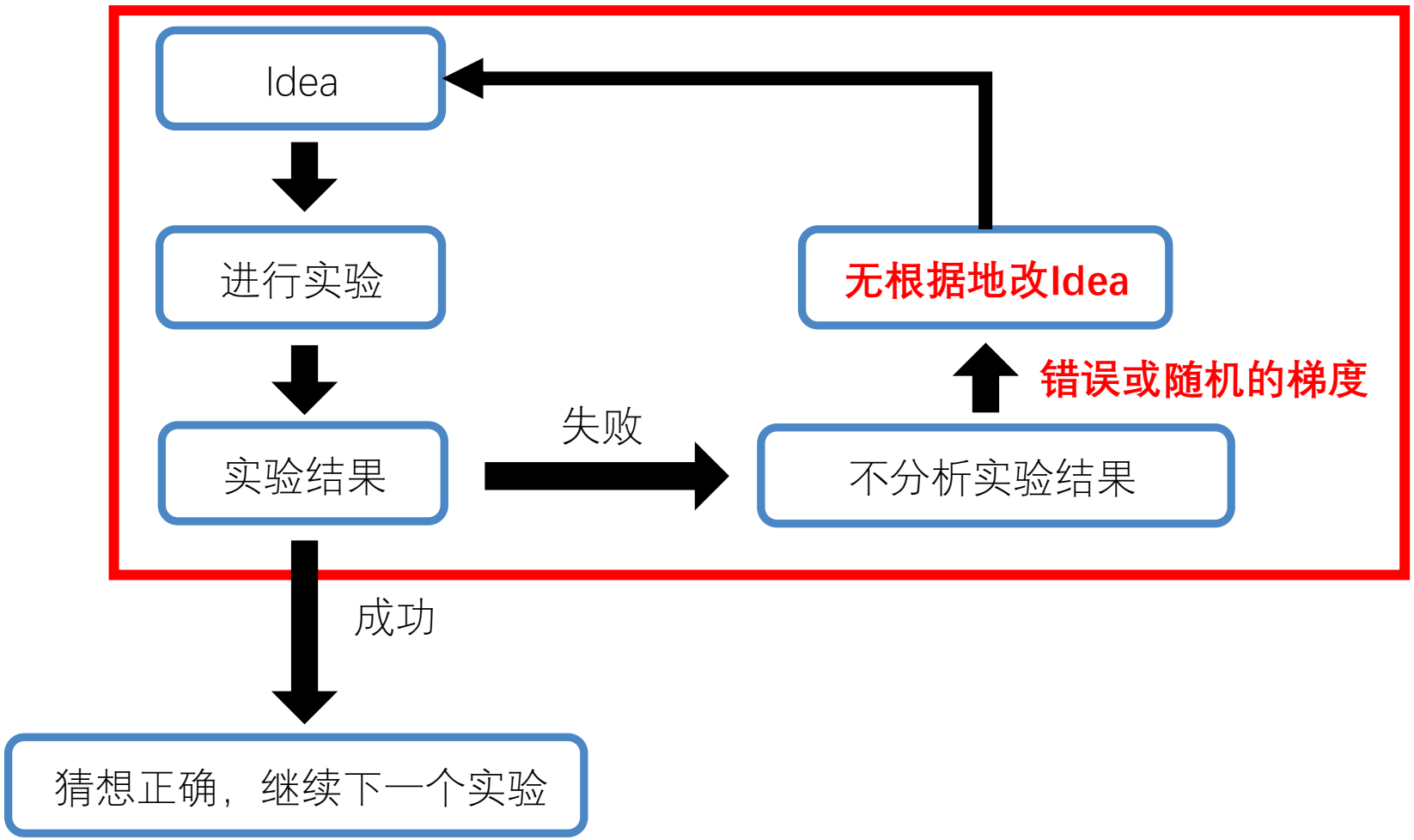
2.2 改进解法、畅想更多解法

- 通过实验的反馈优化Idea → 核心步骤：**分析实验不work的原因**



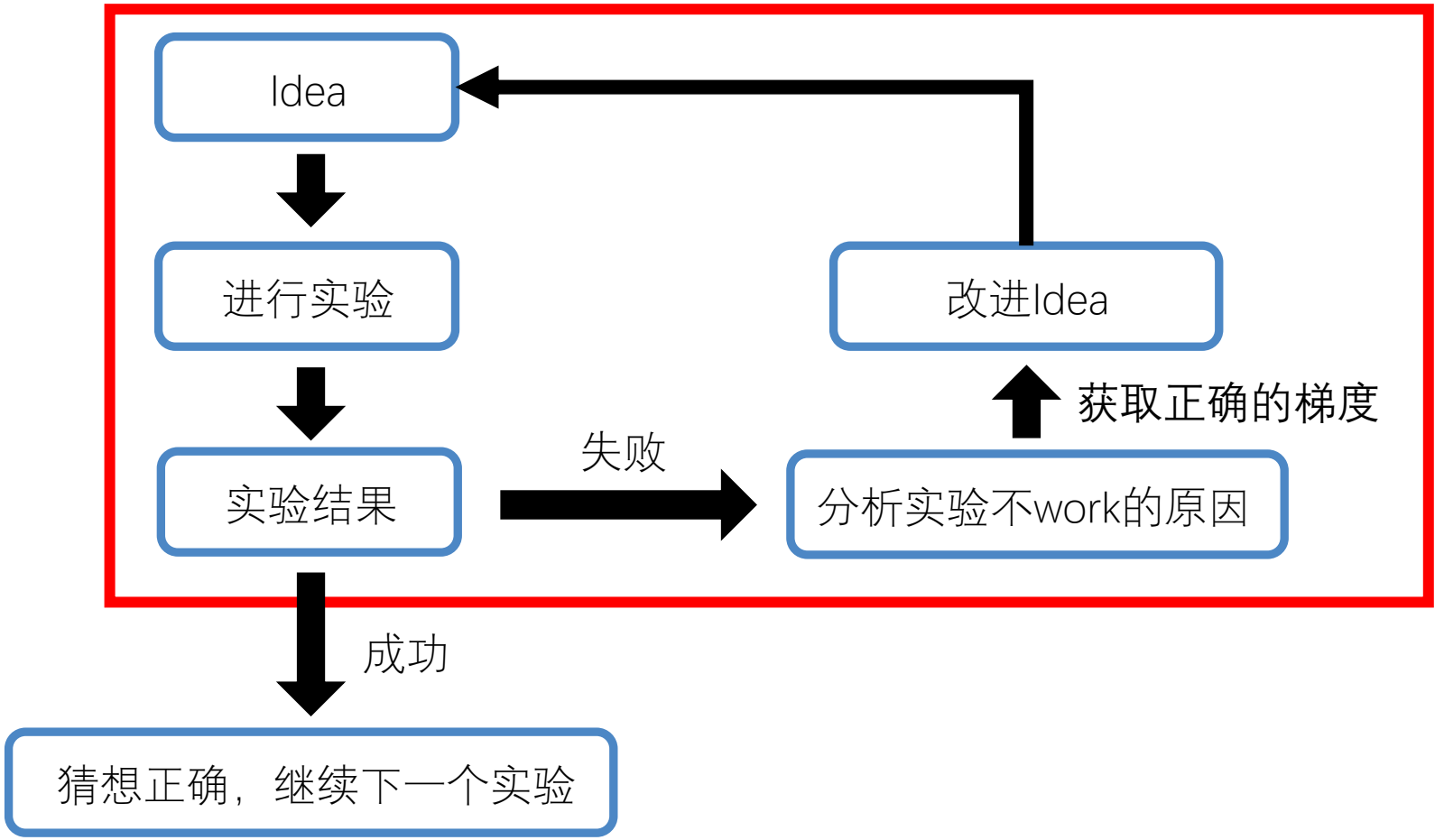
2.2 改进解法、畅想更多解法

- 收敛较慢的Idea优化过程



2.2 改进解法、畅想更多解法

- 这一部分的循环速度决定了做Project的时间：1天5次循环 vs. 1天1次循环，前者做一个月等于后者做五个月。

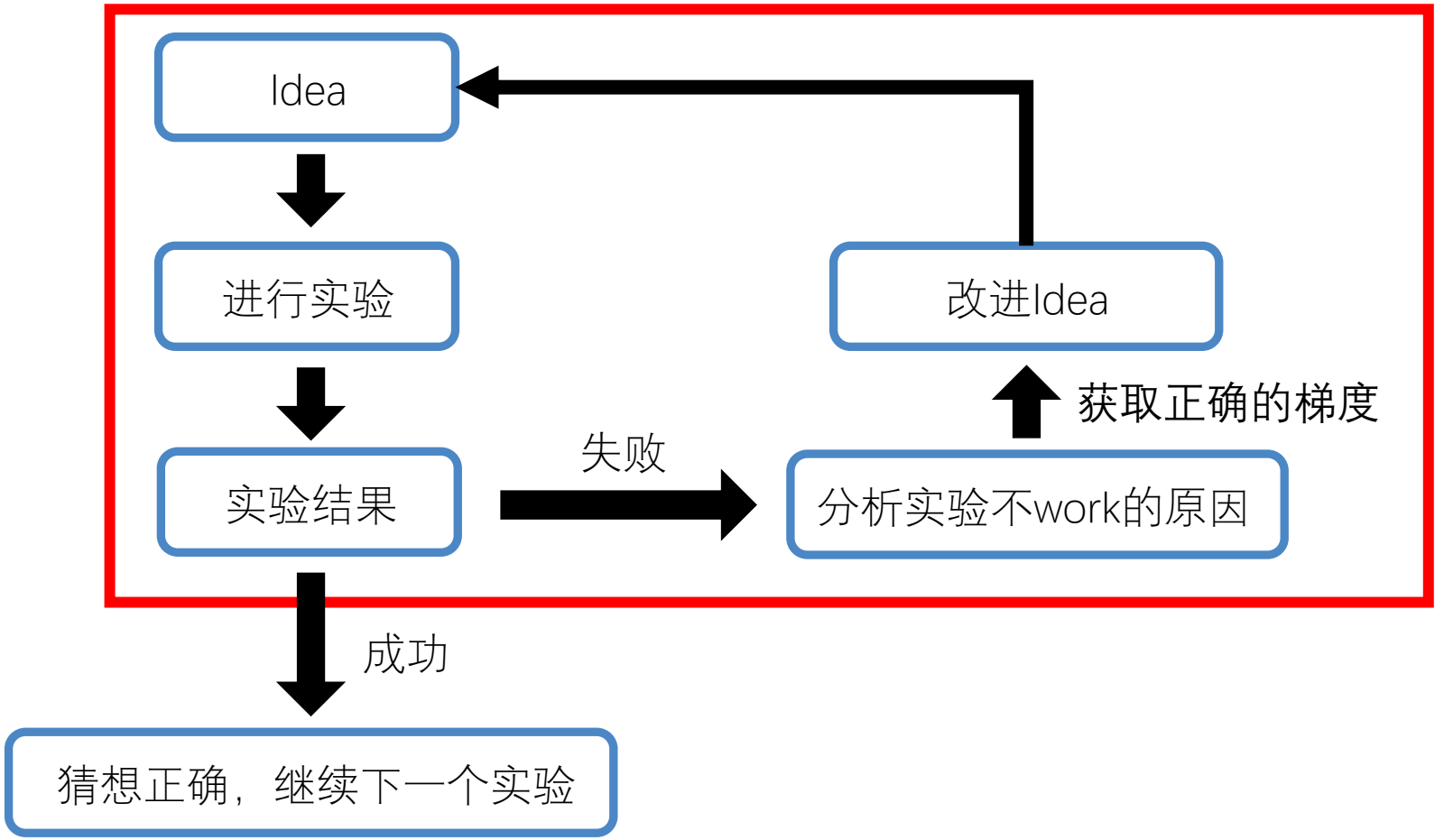


2.2.1 做Project速度慢会面临的问题

Project被scoop（被抢先发布），导致之前投入的时间白费。

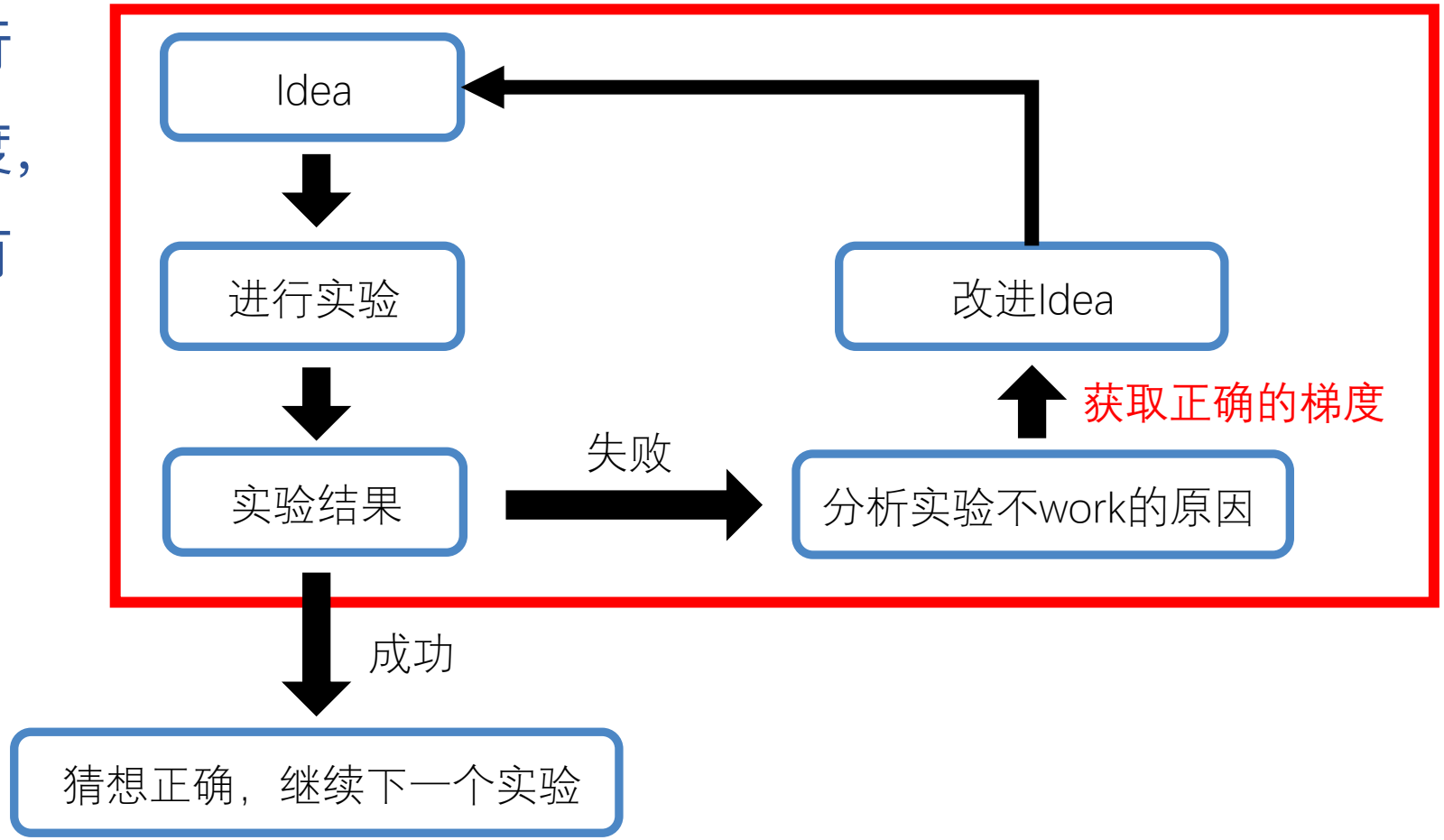
2.2.1.1 如何提升实验迭代速度：实验技巧

- 并行地做实验：同时尝试多个实验



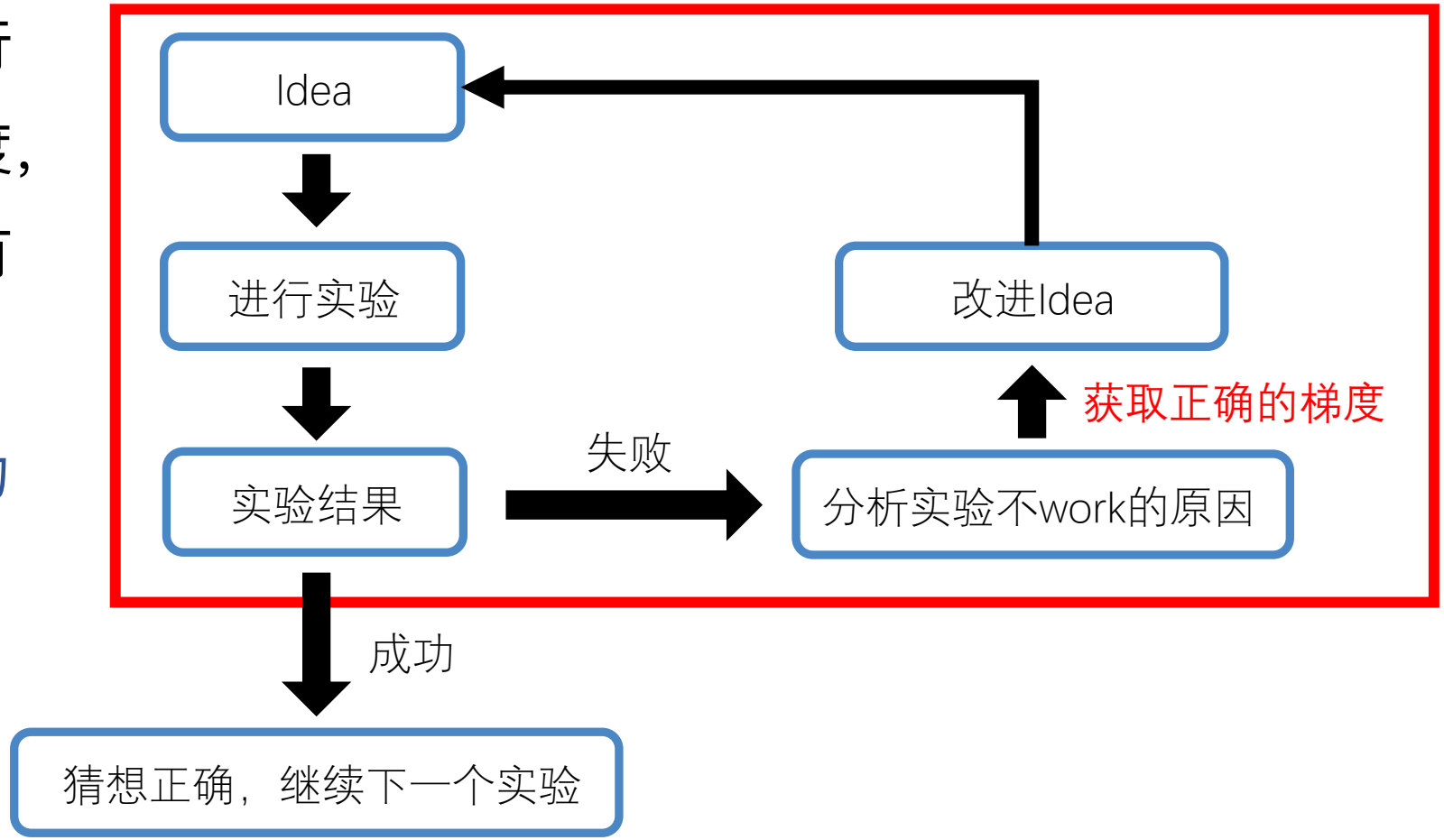
2.2.1.2 如何提升实验迭代速度：获得更好的梯度

- 如何获得更好的梯度：
  - 和导师、同学讨论，进行知识蒸馏，直接获得梯度，或者修正自己的梯度，有效地避免局部最优解



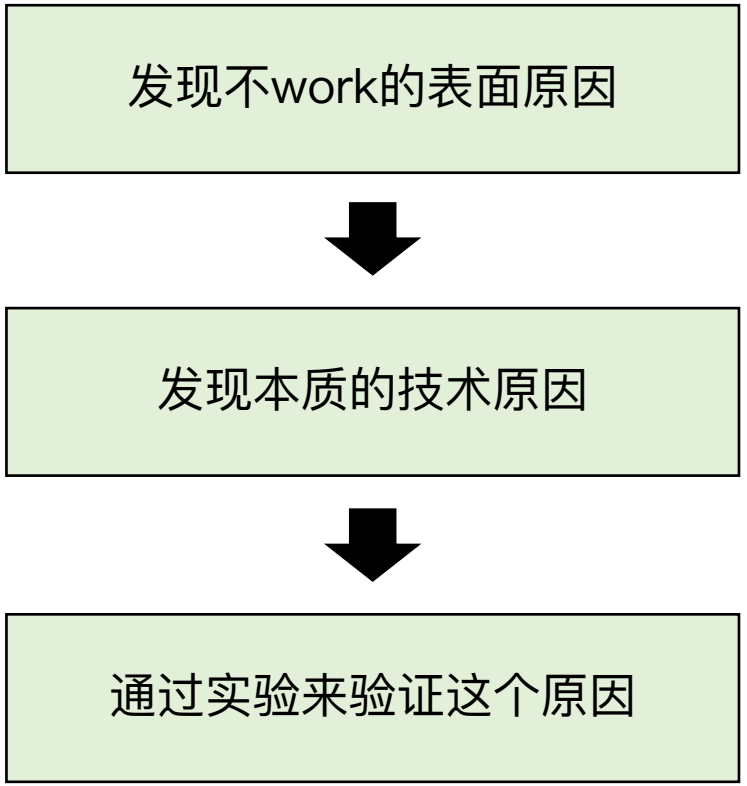
2.2.1.2 如何提升实验迭代速度：获得更好的梯度

- 如何获得更好的梯度：
  - 和导师、同学讨论，进行知识蒸馏，直接获得梯度，或者修正自己的梯度，有效地避免局部最优解
  - 提升分析实验结果的能力



2.2.2 如何分析实验不work的原因

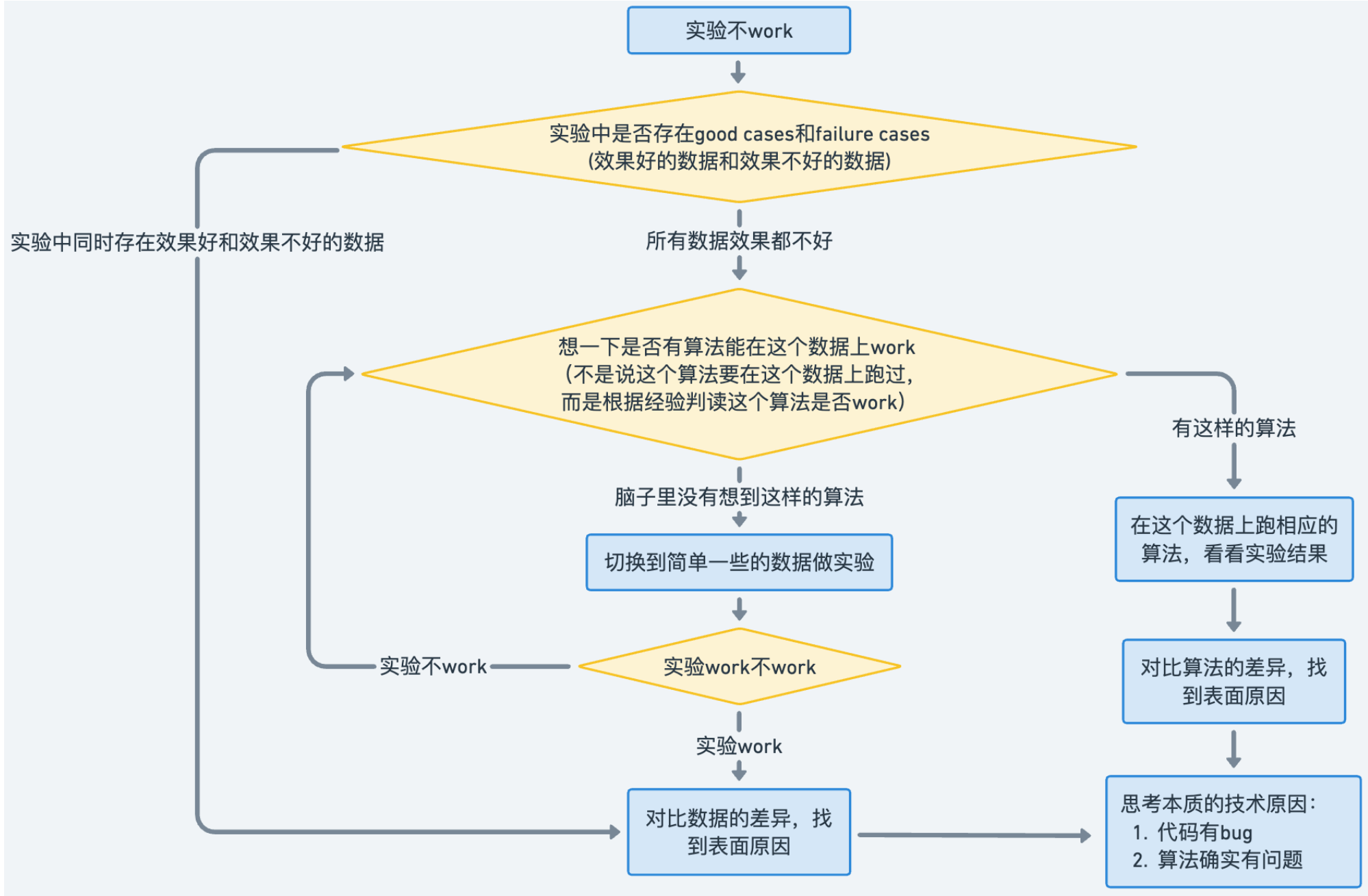
- 表面原因有哪些：
  - 换到某个数据后，效果变得不好了
  - 加了某个module后，效果变得不好了
  - 改了某个参数后，效果变得不好了
  - …
- 本质的技术原因有哪些：
  - 为什么换到某个数据后，效果变得不好了
  - 为什么加了某个module后，效果变得不好了
  - 为什么改了某个参数后，效果变得不好了
  - …



2.2.2.1 如何发现实验不work的表面原因

- 找到work的数据/算法，再对比不work的数据/算法，看看它们有什么不同。

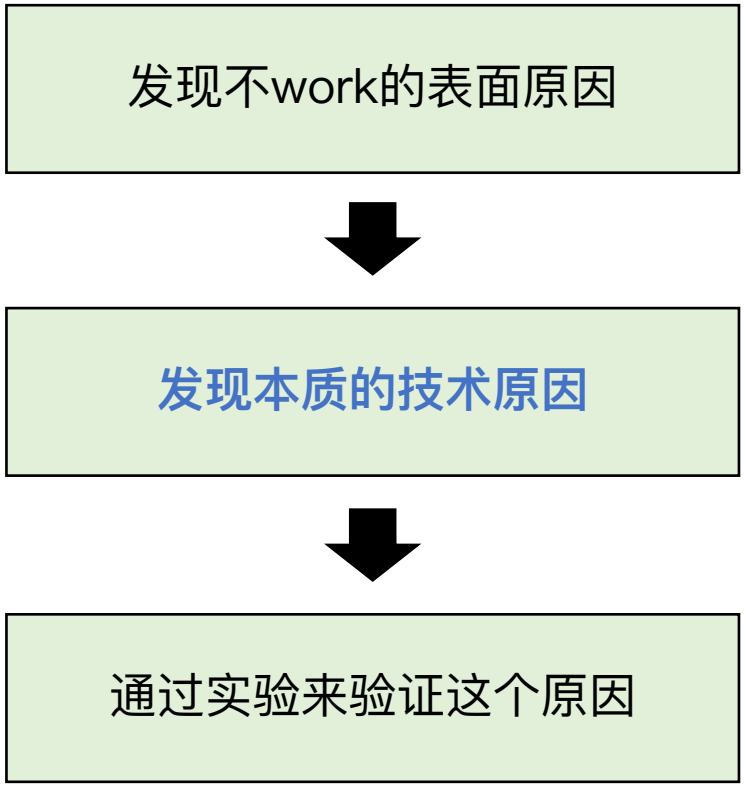
2.2.2.1 如何发现实验不work的表面原因（切勿完全按照这个流程执行，要灵活一些）



2.2.2.2 如何发现实验不work的本质技术原因

列出尽量多的可能的技术原因

- 本质的技术原因有哪些（**思考为什么**）：
  - 为什么换到某个数据后，效果变得不好了
  - 为什么加了某个module后，效果变得不好了
  - 为什么改了某个参数后，效果变得不好了
  - …



2.2.2.2 如何发现实验不work的本质技术原因

一般有两个技术原因：

1. 可能是代码有bug。
2. 可能是“xx算法”在“xx数据”上确实有问题。

   算法有问题的三种可能：超参没设置对、算法缺了几个tricks、算法本身确实不行。

2.2.2.2.1 怎么找算法的问题

一个有效的方法是看相关的论文为什么可以work，看他们使用了什么tricks。

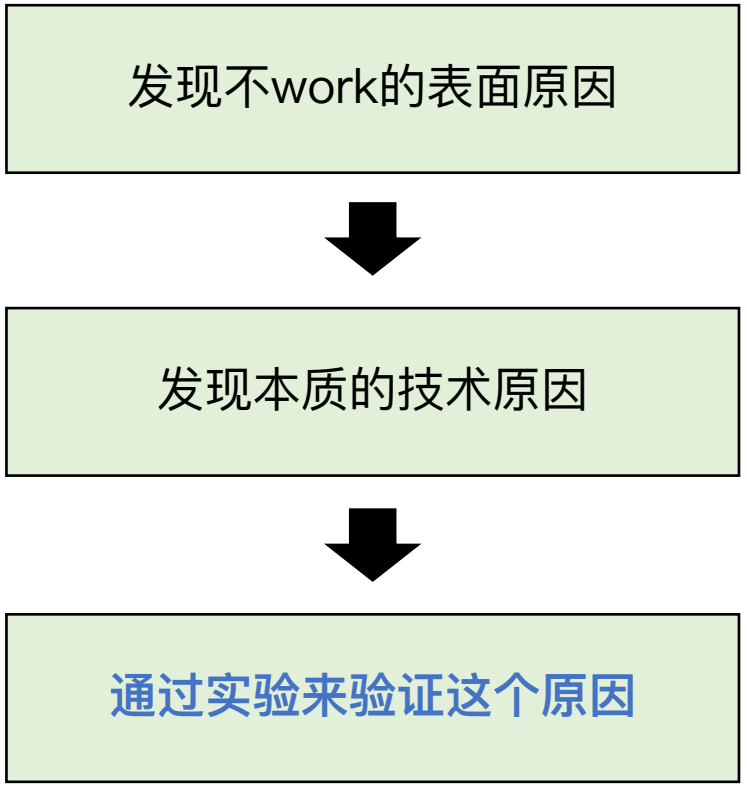
给相关的论文算法做ablation study，看看算法的哪些部分是关键。

有时候，一些算法确实是work的，只是少了点tricks。也就是说，这些牛逼的ideas，单独自己的时候不work，需要加一些tricks才work。（比如，NeRF + positional encoding）

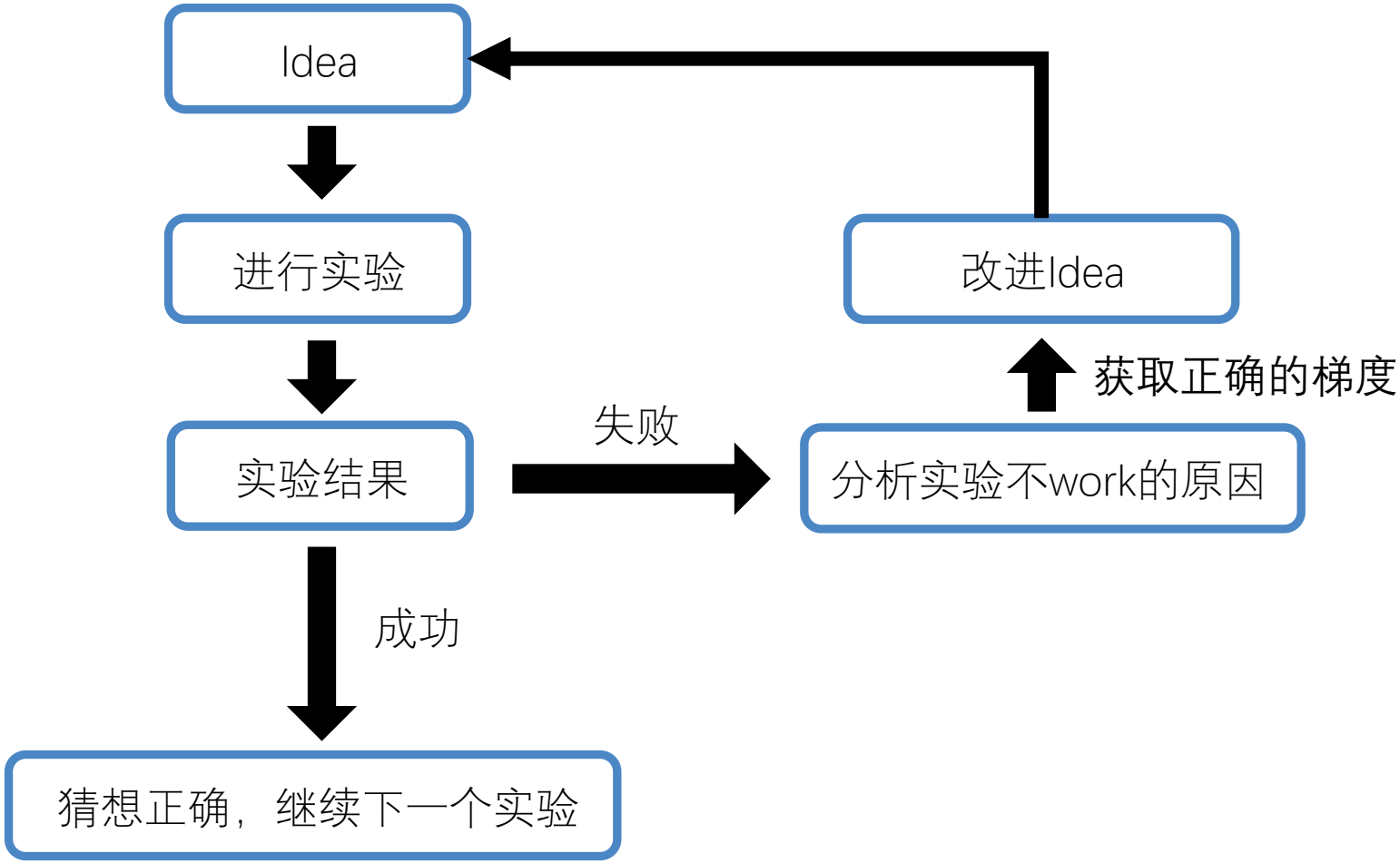
2.2.2.3 通过实验来验证找到的原因

- 列出尽量多的可能的技术原因以后，**通过实验来验证找到的原因。**

- 需要先确定真的是“技术原因”，才能有效地改进算法。



2.2 针对找到的技术原因，改进解法、畅想更多解法



2.3 在真实的数据上，把算法调work

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work

3. 如何写论文

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 论文写作的时间管理
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

3.1 论文写作的时间管理

时间点	论文要写的内容
截稿时间四周前	1. 整理现有的story，包括core contribution、方法的各个模块及其motivation（推荐用脑图整理）。 2. 列出要做的comparison experiments和ablation studies（推荐用脑图整理）。 3. 这一周写一个introduction的初稿。
截稿时间三周前	这一周最好能把方法定下来。 1. 这一周写一个method的初稿。这一周至少method框架定下来了，所以可以把method开始写起来。如果方法的细节还没定下来，就在相应的地方写给\todo{}，先空着，至少把method的框架写出来。 **这周截止，必须把introduction和method的初稿给导师改，不然导师很可能改不完论文**（想象一下，导师最后几天开始改十篇非常不完整的论文，是什么样的地狱体验。如果自己面对这种情况，会是什么心情）。
截稿时间两周前	这一周把experiments, abstract, related work写一个初稿。
截稿最后一周	改论文、画图、做demo

3.2 如何写论文

- 链接：https://github.com/pengsida/learning_research
  总结了常用的论文写作模板

Highlights

1. 如何构思论文Idea
2. 如何开展探索性实验
3. 论文写作模板
4. 各种科研技巧

- 论文标题
- Abstract
- Introduction
- Method
- 论文画图
- Experiments
- Related work
- Conclusion

3.2.1 论文中做什么实验能提升论文的影响力

- 提升人们对论文算法的想象力
  - 尝试更具有挑战性的数据
  - 尝试各种场景下的数据
- 尝试算法对各种下游应用的支撑

3.3 如何review自己的论文：仔细检查论文是否存在被拒的因素

1. Contribution不够（论文没有给读者带来新的知识）	1.1 想解决的failure cases很常见
	1.2 提出的技术已经被well-explored了，该技术带来的performance improvement是可预见的/well-known的
2. 写作不清楚	2.1 缺少技术细节，不可复现
	2.2 某个方法模块缺少motivation
3. 实验效果不够好	3.1 只比之前的方法效果好了一点
	3.2 虽然比之前的方法效果好，但效果仍然不够好
4. 实验测试不充分	4.1 缺少ablation studies
	4.2 缺少重要的baselines、缺少重要的evaluation metric
	4.3 数据太简单，无法证明方法是否真的work
5. 方法设计有问题	5.1 实验的setting不实际
	5.2 方法存在技术缺陷，看起来不合理
	5.3 方法不鲁棒，需要每个场景上调超参
	5.4 新的方法设计在带来benefit的同时，引入了更强的limitation，导致新方法的收益为负

一个research project包含哪些流程

1. 如何想Idea	1.1 规划Project要解决的任务
	1.2 发现任务中需要解决的technical challenge、failure cases
	1.3 构思解决failure cases的方法
2. 如何做实验	2.1 设计简单实验快速验证解法的正确性
	2.2 在简单数据上，改进解法、畅想更多解法
	2.3 在真实数据上，做实验把解法调work
3. 如何写论文	3.1 根据论文的截止日期设置自己论文重要的几个截止时间
	3.2 写论文
	3.3 Review自己的论文
	3.4 改论文

浙江大学
ZHEJIANG UNIVERSITY

谢谢！

报告人：彭思达

GitHub Repo: https://github.com/pengsida/learning_research