



# 2024 AI+研发数字峰会

AI+ Development Digital summit

AI驱动研发迈进数智化时代

中国·上海 05/17-18

## 智能化研发在百度的落地

张立理 百度

# 科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



 **K+峰会**  **上海站**

**K+ 全球软件研发行业创新峰会**

时间: 2024.06.21-22

 **K+峰会**  **敦煌站**

**K+ 思考周®研习社**

时间: 2024.10.17-19

 **K+峰会**  **香港站**

**K+ 思考周®研习社**

时间: 2024.11.10-12



K+峰会详情



 **AiDD峰会**  **上海站**

**AI+研发数字峰会**

时间: 2024.05.17-18

 **AiDD峰会**  **北京站**

**AI+研发数字峰会**

时间: 2024.08.16-17

 **AiDD峰会**  **深圳站**

**AI+研发数字峰会**

时间: 2024.11.08-09



AiDD峰会详情



## 张立理

百度前端架构师

---

百度前端架构师，百度前端CMC主席，技术组织委员会Web方向负责人。

2023年起参与百度智能开发工具Comate的IDE端架构、模型提升等方面工作，对续写、生成、问答等软件开发环节的大模型应用有丰富经验

# 目录

## CONTENTS

1. 背景与效果
2. 构造优秀的智能研发助手
3. 智能化的进一步发展与应用
4. 思考、总结及展望

# PART 01

## 背景与效果

**AI 就像 100 年前的电力革命，  
正在改变各行各业**

— 吴恩达

# ► 以往的研发提效工作

## 环境

跨OS开发困难

办公电脑配置低，开发体验差

环境在线化

开发环境复杂，配置成本高

环境不一致，异常难复现

## 协作

工作涉及多平台，切换打断严重

上下游沟通频繁

IDE功能集成

沟通反馈周期长，不易记忆

## 编码

相似代码编写效率低

重构潜在风险高

??

可复用片段不易发现

编码过程缺少流畅感

# ▶ 大模型横穿出世，各生态迅速发展

节点	事件
2020年6月	GPT3发布
2021年7月	GitHub Copilot发布
2022年9月	Whisper发布，LLM正式引入语音转文本
2022年11月	ChatGPT发布
2023年3月	GPT 3.5 Turbo发布，成本降低90%
2023年3月	GPT4发布，极大提升置信度能力
2023年3月	Auto-GPT发布，提示词工程的一次伟大演示
2023年4月	GitHub Copilot产品矩阵丰富，包括Docs、PR、CLI



## 帮你想

调用、设计

需求澄清  
任务分解  
代码解释  
技术问答

## 帮你写

编码、实现

实时续写，注释生成代码  
描述生成代码，生成注释  
接口生成代码  
生成单元测试

## 帮你改

检测、优化

代码Debug，缺陷自动修复  
生成测试用例，代码优化  
智能评审，CI错误诊断  
安全检查与修复，智能CLI

# ▶▶ 落地效果 - 内部

80%

用户渗透

46%

采纳率

27%

代码占比

# ▶▶ 落地效果 - 外部

**1W+**

注册企业数

**49%**

采纳率

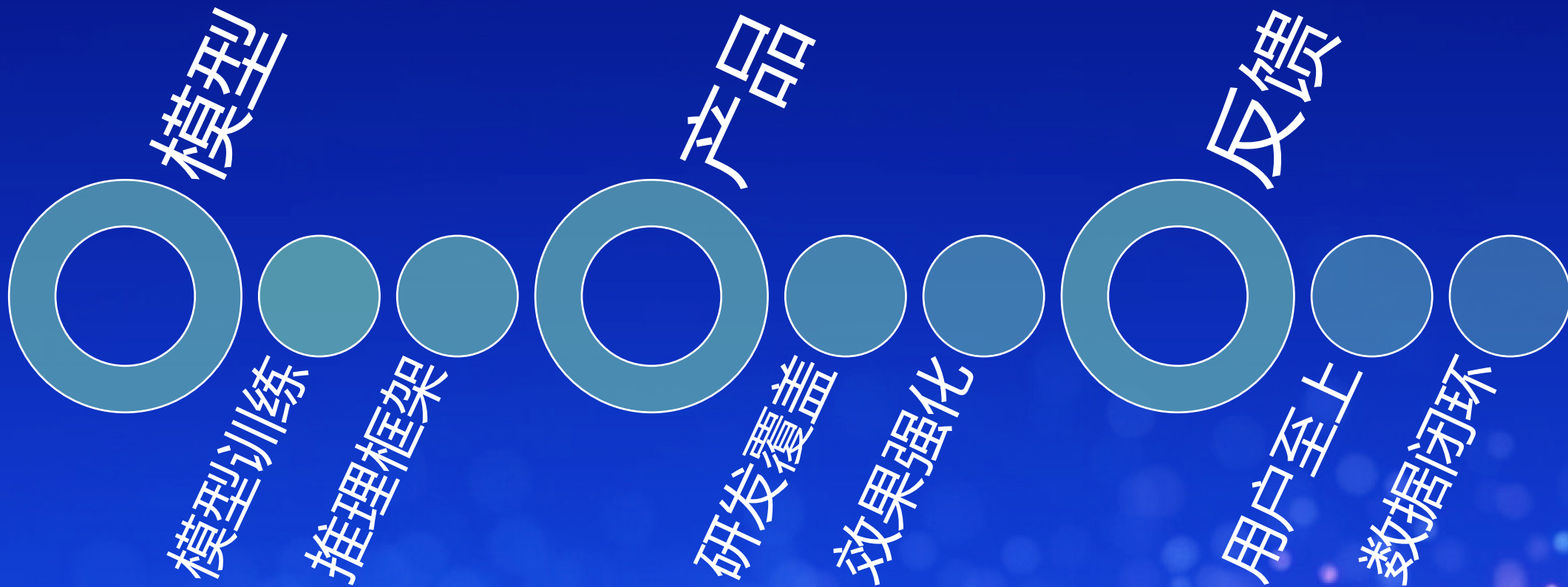
**10+**

私有化部署合作

## **PART 02**

# **构造优秀的智能研发助手**

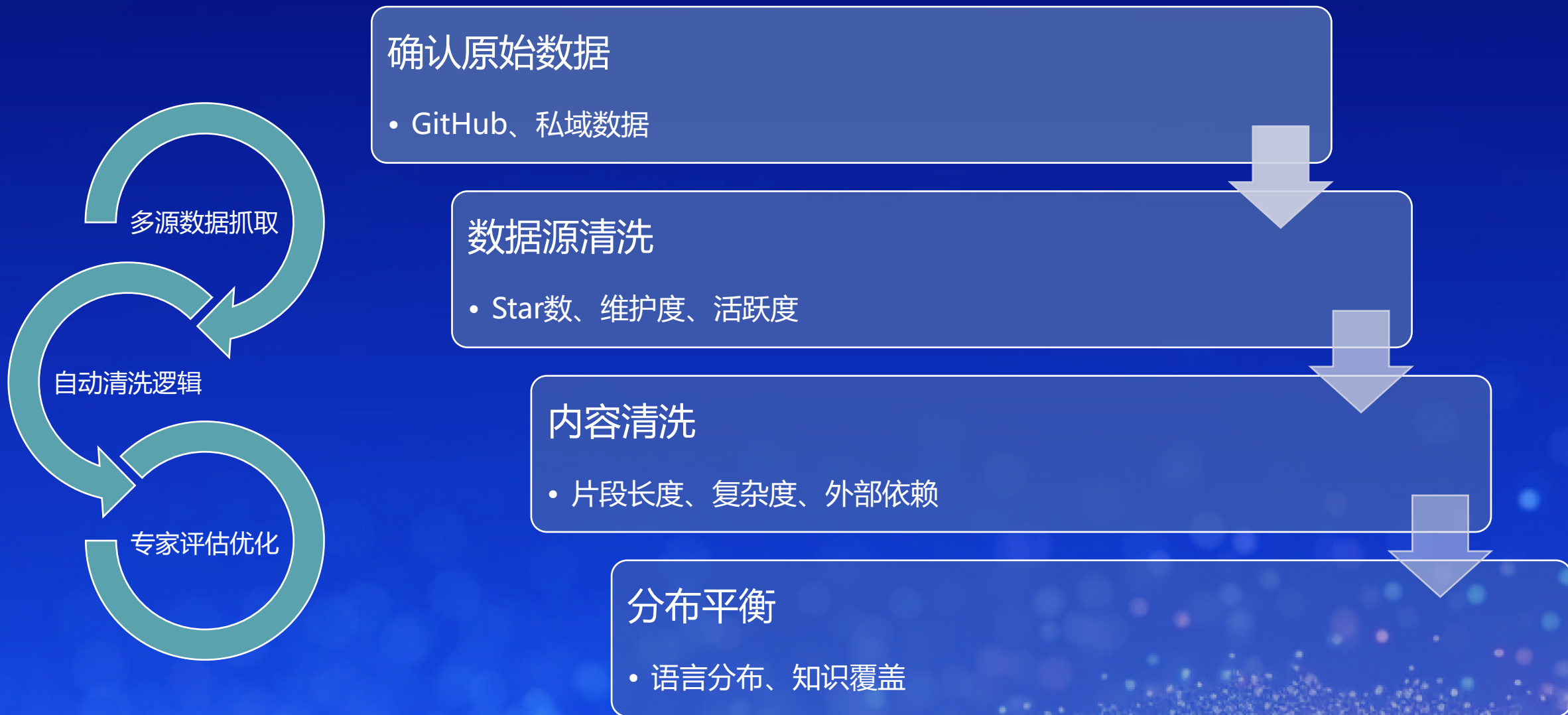
# ▶ 多环节打造优秀效果



# ▶▶ 构建强力的代码模型

- 模型效果依赖于训练数据质量
- 编码与传统文本生成形态存在差异
- 研发工作对生成质量以外因素有要求

# ▶▶ 问题1 – 代码 ≠ 高质量代码



# ▶ 问题2 - 编码 ≠ 尾部生成

```
import React, { useState, useEffect } from 'react';

interface Weather {
  ...
}

const WeatherCard: React.FC = () => {
  const [weather, setWeather] = useState<Weather | null>(null);

  useEffect(() => {
    // data mock
    setWeather({
      name: "London", main: { temp: 15 }, weather: [{ description: "Clear Sky" }]
    });
  }, []);
  if (weather === null) {
    return <div>Loading...</div>;
  }
  return (
    <div>
      <h1>{weather.name}</h1>
      <h2>{weather.main.temp}°C</h2>
      <h3>{weather.weather[0].description}</h3>
    </div>
  );
};

export default WeatherCard;
```

```
interface Weather {
  ...
  temperature: number,
  condition: string,
  location: string,
  humidity: number
}
```

尾部续写逻辑

```
interface Weather {
  ...
  name: string;
  main: {
    temp: number;
  };
  weather: {
    description: string;
  }[];
}
```

中间插入 (FIM)



## ▶ 问题3 – 质量好 ≠ 愿意等

编码过程是一个**连续、专注**的过程，打断、等待对持续编码的工程师是负面反馈  
代码补全（续写）的**性能诉求**相比普通文本生成更高  
在保证质量的前提下，需要关注**响应速度**

基于PaddlePaddle，推理跟上编码速度工，优先保障使用体验

- 端到端响应 < 600ms

适配多种推理硬件，确保用户环境可部署、使用体验到位

- 4090、国产硬件、一体机

SaaS推理服务，弹性资源，确保响应速度

- SaaS版本免费使用

## 高质量数据

- 广泛数据来源
- 精细清洗挑选
- 专家参与建设

## 符合编码结构

- 支持FIM模式
- 探索前后文比例
- 代码依赖学习

## 体验性能优先

- 高性能打理优化
- 高效模型压缩
- 适配多类硬件

# ▶▶ 产品是人与模型的中介

- 人机界面是产品，而非模型
- 开发者的语言、IDE多样性
- 产品形态需要与编码工作紧密整合
- 研发现场对代码准确、高质有决定性作用
- 代码安全性是企业的重要考量

## 支持100+语言

尤其在C/C++、Java、Python、Go、JavaScript、TypeScript等主流语言表现出色

## 主流IDE



# 与实践紧密结合的产品逻辑



# ▶ 提供丰富的编码期能力

```
func prefixZeroClear(input string) string {
    for i := 0; i < len(input); i++ {
        if input[i]!=' ' {
            return input[:i]
        }
    }
    return ""
}
```

```
// GUNZIP 解压缩文件
增强生成
func GUNZIP(filename string) (io.ReadCloser, error) {
    gzFile, err := os.Open(filename)
    if err != nil {
        return nil, err
    }
    defer gzFile.Close()
    r, err := gzip.NewReader(gzFile)
    if err != nil {
        return nil, err
    }
    return ioutil.NopCloser(r), nil
}
```

自动代码续写

wangchuqing

为下面的函数生成单元测试:

```
func intToBytes(i int32) []byte {
    bytes := make([]byte, 4)
    binary.BigEndian.PutUint32(bytes, uint32(i))
    return bytes
}
```

Comate

Go  采纳 |  复制

```
func TestHeaderParse(t *testing.T) {
    tests := []struct {
        name          string
        headerStr     string
        expectedHeader Header
        expectedErr   bool
    }{
        {
            name:          "Empty header",
            headerStr:     "",
            expectedHeader: Header{
                expectedErr: false,
            },
        },
        {
            name:          "Bad magic code",
            headerStr:     "\xff\xff",
            expectedHeader: Header{
                expectedErr: true,
            },
        },
    }
    for _, test := range tests {
        t.Run(test.name, func(t *testing.T) {
            header, err := ParseHeader(test.headerStr)
            if err != test.expectedErr {
                t.Errorf("ParseHeader(%s) = %v, %v; want %v, %v", test.headerStr, header, err, test.expectedHeader, test.expectedErr)
            }
        })
    }
}
```

代码生成单测

wangchuqing

为下面的函数生成注释:

```
def handle_bug(self, oneBug, bug_level):
    """
    bug过滤与信息补充
    :param oneBug:
    :param bug_level:
    """
```

Comate

Python  采纳 |  复制

对bug进行过滤和信息补充

Args:

- oneBug (dict): 待处理的bug
- bug\_level (str): bug的等级
- bug\_set (set): bug集合

Returns:

- None: 没有返回值, 直接修改传入的bug数

代码生成注释

wangchuqing

python遍历文件夹的所有子文件和子目录

```
MagicCode == nil || len(MagicCode) != 4 {
    return fmt.Errorf("MagicCode should be 4 bytes")
}
```

Comate

以下是使用Python遍历文件夹的所有子文件和子目录的示例代码:

Python  复制

```
import os

def traverse_directory(path):
    for root, dirs, files in os.walk(path):
        # 遍历子目录
        for dir in dirs:
            print(os.path.join(root, dir))
        # 遍历子文件
        for file in files:
            print(os.path.join(root, file))
```

这个函数使用os.walk()函数来遍历指定的路径下的所有子目录和子文件。在每次迭代中,os.walk()会返回当前目录的路径、子目录列表和

自然语言生成代码

# ▶ 基于研发现场的知识增强

私有框架、库，项目内类、函数，在模型的知识中**不存在**，但在编码实践中至关重要  
 使用SFT强化模型**成本高昂**，具备**滞后性**，无法真正解决工程实践中的问题  
 通过**知识增强** (RAG)，可以通过模型上下文有效解决私域知识应用问题  
 对于代码续写，IDE**现场**能够提供多样化的知识来源

## Neighbour



- 基于文本相似性
- 源于当前打开、近期打开文件
- 适用广泛场景

## Dependency



- 基于LSP进行定义查询
- 提供准确的依赖签名
- 需源码中有明确的导入引用

## Embedding



- 基于向量相似度索引
- 有更好的语义准确性
- 更适用注释引导生成等场景

# ▶ Neighbour加强推理

```
TS index.tsx 1, U
> TS index.tsx > WeatherCard > useEffect() callback > fetchWeather > response > params
17  ... useEffect(() => {
18  ...     const fetchWeather = async () => {
19  ...         try {
20  ...             const response = await axios.get<Weather>(
21  ...                 'http://api.openweathermap.org/data/2.5/weather'
22  ...             , {
23  ...                 params: {
24  ...                     q: 'London',
25  ...                     appid: 'key-abcabcabcabc'
26  ...                 }
27  ...             });
28  ...             setWeather(response.data);
29  ...         } catch (error) {
30  ...             console.error('Failed to fetch weather:', error);
31  ...         }
32  ...     };
33  ...
34  ...     fetchWeather();
35  ... }, []);
36
```

```
.env
src > components > .env
1  APPID = key-abcabcabcabc
```



# ▶ Dependency加强推理

```
TS index.ts 1, U • TS Coupon.ts U TS Product.ts U •
```

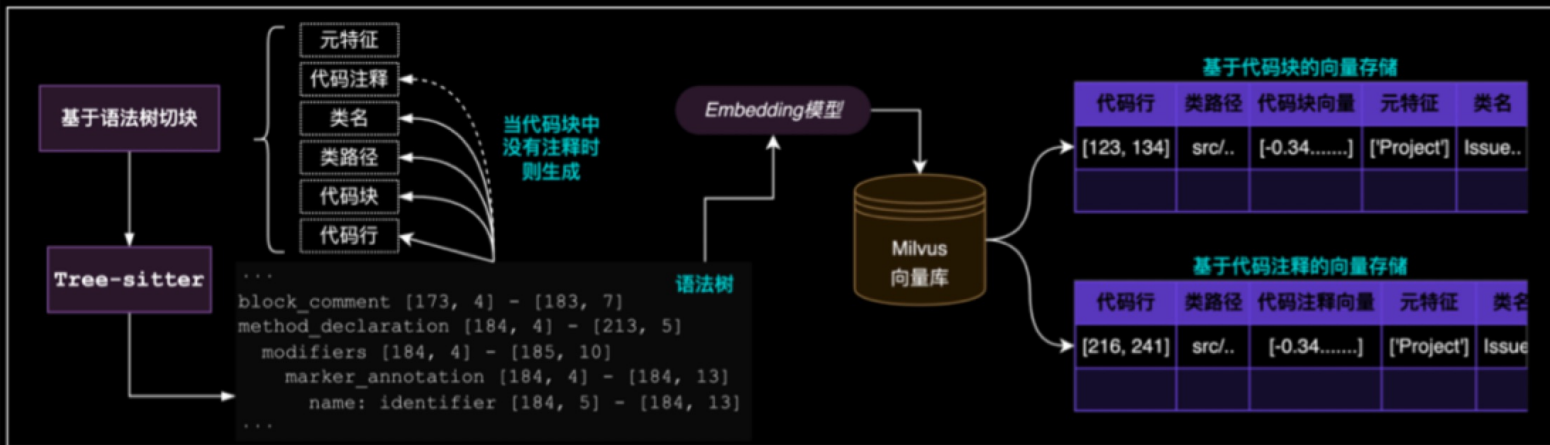
```
src > components > TS index.ts > Order > addProduct
1  import {Product} from "../Product";
2
3  export class Order {
4      ... products: Product[];
5      ... total: number;
6
7      ... constructor() {
8          ... this.products = [];
9          ... this.total = 0;
10     ... }
11
12     ... addProduct(product: Product, coupon?: Coupon) {
13         ... this.products.push(product);
14         ... this.total += product.getPrice(coupon)
15     ... }
16
17 }
18
```

```
src > components > TS Product.ts > Product
1  export class Product {
2      ... private name: string;
3      ... private description: string;
4      ... private price: number;
5
6      ... constructor(name: string, description: string, price: number) {
7          ... this.name = name;
8          ... this.description = description;
9          ... this.price = price;
10     ... }
11
12     ... getPrice(coupon?: Coupon): number {
13         ... if (coupon?.discountAmount) {
14             ... return this.price - coupon.discountAmount > 0 ? this.price - coupon.discountAmount : 0;
15         ... }
16         ... if (coupon?.discountRate) {
17             ... return this.price * (1 - coupon.discountRate);
18         ... }
19         ... return this.price;
20     ... }
21
```

# ▶ Embedding向量优化

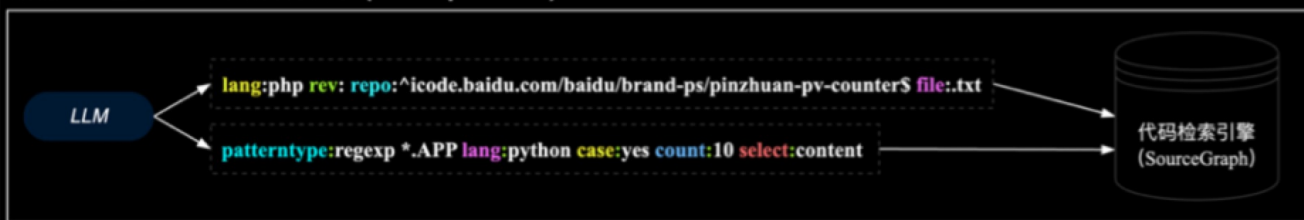
## 路径一：基于语法树的向量检索

重点在于自然语言检索代码，检索『代码做什么』，用于分析技术目标时早期检索



## 路径二：基于LSIF的结构化检索 (NL2Expression)

重点在于了解代码架构，偏向于关键词检索，用于分析技术时早期检索



## 路径三：基于本地IDE Symbol检索



重点在于调用逻辑检索，用于Agent生成代码期间读取相关代码

基于不同语言语法定制向量化策略，能够有效提升召回质量  
Embedding加强用于对话等形式的代码生成

# ▶ Embedding加强效果

Paddle框架

• 60.5 -> 81.5

百度小程序开发

• 70 -> 90

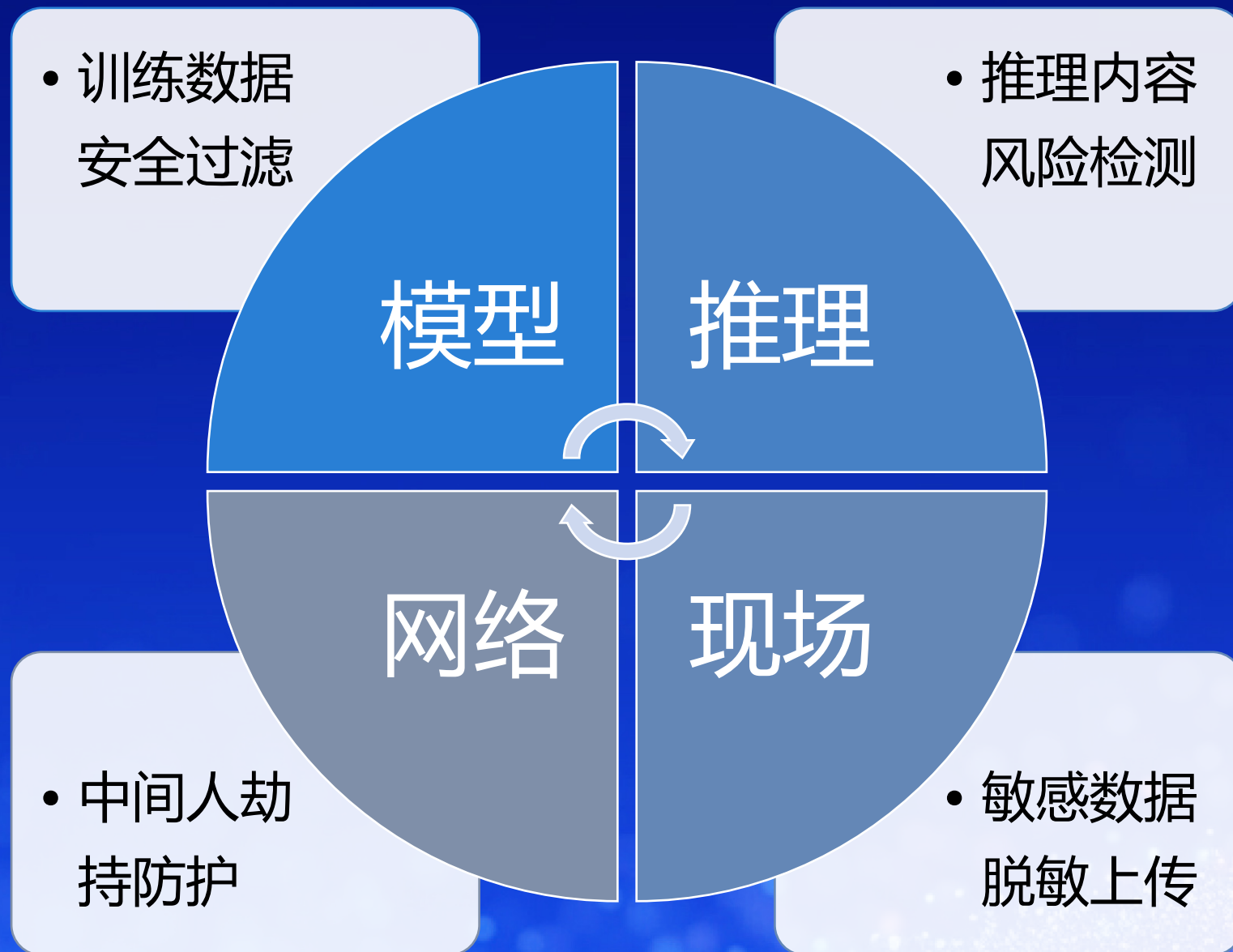
某内部函数计算框架

• 40 -> 80

某内部GO服务端框架

• 0 -> 62.5

# ► 独有多层加固，确保代码数据安全



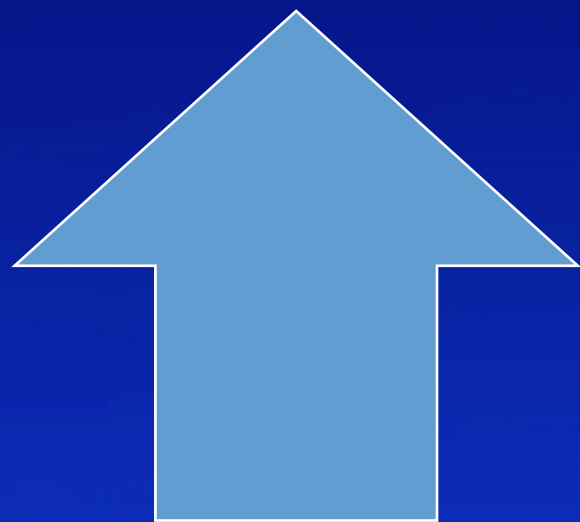
# ► 基于反馈、用户优先的产品飞轮



# PART 01

# 智能化的进一步发展与应用

# ▶▶ 继续深入智能化研发



## 更智能

- 大规模代码生成
- 跨研发阶段影响力



## 更易用

- 连接三方能力
- 适配团队基建

# ▶▶ AutoWork – 从需求到代码

- 开发者仅需要把需求文档写好，剩下的由 AutoWork 自主思考、拆解需求并执行任务，完成代码的生成
- 基于文心大模型思维链能力，像智能体一样思考和执行任务，听懂人的需求，顺序执行多个步骤
- 利用RAG代码智能检索技术，将知识库信息补充到大语言模型中，大语言模型在用户提问时，检索这些知识库信息来增强其生成的回答
- 专门为编程领域设计研发了 RAG 智能代码检索技术，能够让用户从代码库中检索可能回答问题的最相关内容，从而解决 LLM的幻觉等问题





niuwanpeng  
帮我梳理当前代码架构

@AutoWork

思考过程

- 分析 解析您的目标
- 检索 帮我梳理当前代码架构
- 生成 构建您的答案

搜索结果

当前代码架构梳理如下:

- 目录结构:
  - `Dockerfile`: Dockerfile用于构建Docker镜像。
  - `README.md`: 文档文件, 描述了项目的基本信息、代码结构、如何本地测试等。
  - `knowledge`: 主要业务逻辑代码存放的目录。
    - `auth`: 鉴权相关逻辑。
    - `builder`: 索引构建相关逻辑, 包含多个builder模块, 如 `base_builder.py` 等。

停止生成

全库分析结构

niuwanpeng  
参考Ernie Bot 4.0的API帮我补全当前文件的代码

@AutoWork

思考过程

- 分析 解析您的目标
- 检索 参考Ernie Bot 4.0的API帮我补全当前文件的代码
- 生成 构建您的答案

搜索结果

根据你的问题, 我理解你需要使用Python来实现与Ernie Bot 4.0 API的交互。由于Ernie Bot 4.0是百度提供的服务, 我们需要按照其官方开发文档来实现。这里我将提供一个基础的Python代码框架, 用于获取access token并调用Ernie Bot 4.0的API。首先, 我们需要在代码中实现 `generate_access_token` 函数, 用于获取access token。然后, 我们需要在Ernie类中实现 `_call` 方法, 用于调用Ernie Bot 4.0的API。以下是补全后的代码:

```
Python
```

停止生成

关联私域知识生成代码

markdown\_builder.py

Comate

Comate助理识别到该文件中 `file_splitter_by_filepath` 函数需要添加注释, 即将进行分析...

经过分析, 当前代码片段需要增加函数注释, 优化后的代码如下:

显示变更  采纳  复制

```
def file_splitter_by_filepath(self, filepath, chunk_size, chunk_overlap: int = 0, *kwargs):
    """
    根据文件路径切分文档。

    Args:
        self (BaseBuilder): BaseBuilder实例。
        filepath (str): 需要切分的文件路径。
        chunk_size (int, optional): 每个切片的长度, 默认为chunk_overlap (int, optional): 切片之间的重叠部分
        **kwargs (dict, optional): 可选参数, 包括:
            extract_path (bool, optional): 是否提取相对路径
            relative_path (str, optional): 相对路径, 默认

    Returns:
        List[Document]: 返回切分后的List[Document]列表, 其中metadata包含'Filename'字段, 值为filepath或者提

    Raises:
        None.
    """
    docs = self.markdown_splitter(filepath=filepath, filename, _ = os.path.splitext(os.path.basename(filepath))
    for doc in docs:
        filename = kwargs.get('relative_path') if kwargs.get('relative_path', '') else filepath
        doc.metadata = {
```

助理模式陪伴编程

**4W**

索引代码库

**1.2W**

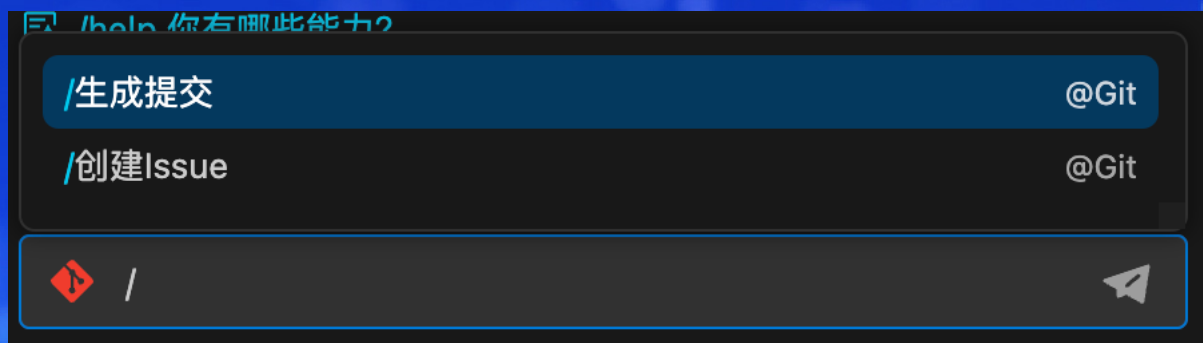
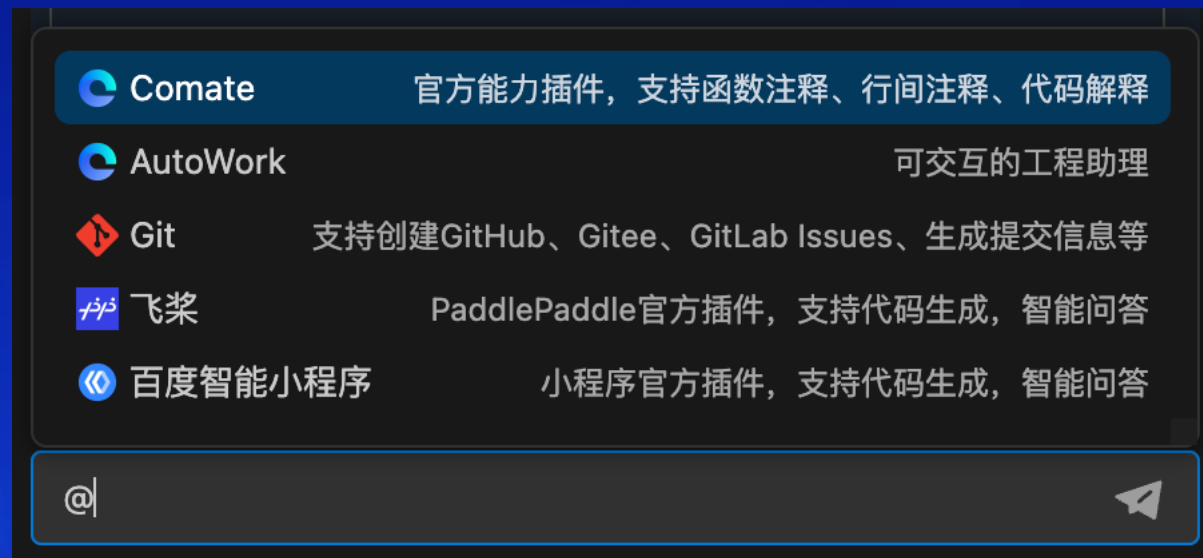
周代码采纳

**40%+**

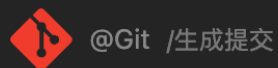
采纳率

# ▶ 开放平台 – 让产品适应组织个人

- 通过开放性，可以更好的让 Comate 来适应不同组织和个人，取得更好的智能化效果
- 通过 Comate 插件这一使用形态，让平台能力直接触达开发人员第一工作现场
- 无需重复建设即可快速大模型能力，无限扩充平台场景
- 让 Comate 更适配团队的业务知识，适配团队规范、固化团队流程



# ▶ 连接三方能力 - Git



@Git /生成提交

正在分析本项目的功能用途.....

正在分析文件 `.gitignore` 的修改内容。

正在分析文件 `.comate/plugin.json` 的修改内容。

正在分析文件 `README.md` 的修改内容。

正在获取项目中的Issue列表.....

使用 #8 新增代码覆盖率目录到`.gitignore`，以及更新`README.md`文件 与当前提交绑定

正在生成提交信息.....

以下是建议的提交信息：

更新`.gitignore`与`README`，关联#8

你也可以直接使用下方的Shell命令提交代码：

Shell

查看变更

采纳

复制

新建文件

插入到终端中

```
git commit -m '更新.gitignore与README，关联#8'
```

## 按需生成Issue标题与内容 调用三方平台API快速创建，无需开发者切换平台



@Git /创建Issue

正在分析本项目的功能用途.....

正在分析文件 `.gitignore` 的修改内容。

正在分析文件 `.comate/plugin.json` 的修改内容。

正在分析文件 `README.md` 的修改内容。

建议标题：

"代码库功能增强：增强安全性配置，优化文档与代码覆盖率报告管理"

正在生成Issue内容.....

建议内容：

修改目标：

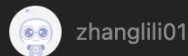
1. 改进代码库的管理效率，通过更新 `.gitignore` 文件来减少不必要的文件干扰。
2. 提高项目安全性，通过 `.comate/plugin.json` 文件配置插件的安全性。
3. 优化项目文档，通过更新 `README.md` 文件来更好地引导社区成员复现和解决问题。

修改内容：

1. 在 `.gitignore` 文件中增加了忽略目录，以确保代码覆盖率报告文件（如由测试工具生成的覆盖率报告）不会被错误地提交到代码库中。这有助于保持代码库的整

通过分析代码更新生成提交信息  
学习历史提交信息规则，生成符合规范的信息  
匹配三方Git平台现有Issue，让代码与需求关联

# ▶ 开放平台 – 让团队知识直达一线



zhanglili01

我想修改系统底部导航的内容，加一个“用户中心”链接



@百度智能小程序 / 智能问答

为了修改。这通常是在小程序的配置文件（如 `app.json` 或 `index.json`）中定义的。

- 在底部导航的配置中，你可以看到一个数组，其中包含了每个导航项的配置信息。每个导航项通常包括一个标题（`text`）和一个对应的页面路径（`pagePath`）。
- 要添加一个新的导航项，你需要在数组中插入一个新的对象。这个对象应该包含你想要添加的“用户中心”链接的标题和页面路径。例如：

json

查看变更 采纳 复制 新建文件

```
{
  "text": "用户中心",
  "pagePath": "pages/userCenter/userCenter"
}
```

在上面的示例中，`text` 属性是导航项的标题，`pagePath` 属性是点击该导航项后跳转的页面路径。

- 保存并重新运行你的小程序，你应该能够在底部导航中看到新添加的“用户中心”链接。

请注意，具体的操作步骤可能会因你使用的小程序框架和版本而有所不同。上述步骤是基于一般的小程序开发流程提供的，你可能需要参考你所使用的小程序框架的文档或示例代码来获取更具体的指导。



@飞桨 / 智能问答

在此函数将接收张量 `x`、整数 `dim` 和 `mask_idx` 作为参数，并按照您的描述执行操作。

首先，请确保您已经安装了 PaddlePaddle 框架。然后，您可以使用以下代码：

Python

查看变更 采纳 复制 新建文件

```
import paddle
import paddle.nn.functional as F

def paddle(x, dim, mask_idx):
    # 对输入张量x进行softmax操作，操作维度为axis=dim
    x1 = F.softmax(x, axis=dim)

    # 将softmax的输出x1的第mask_idx列对应元素置为0
    x1 = paddle.set_value(x1, paddle.to_tensor(mask_idx),

    # 对结果的第2维进行求和操作并保持维度，输出为x1_sum
    x1_sum = paddle.sum(x1, axis=1, keepdim=True)

    # 将x1除以x1_sum并返回结果
    result = x1 / x1_sum

    return result
```

# ▶▶ 开放平台 – 每一个人的专属助手



**6.5W**

注册知识

**10+**

扩展插件

**6W+**

周代码生成

# PART 01

## 总结、思考与展望



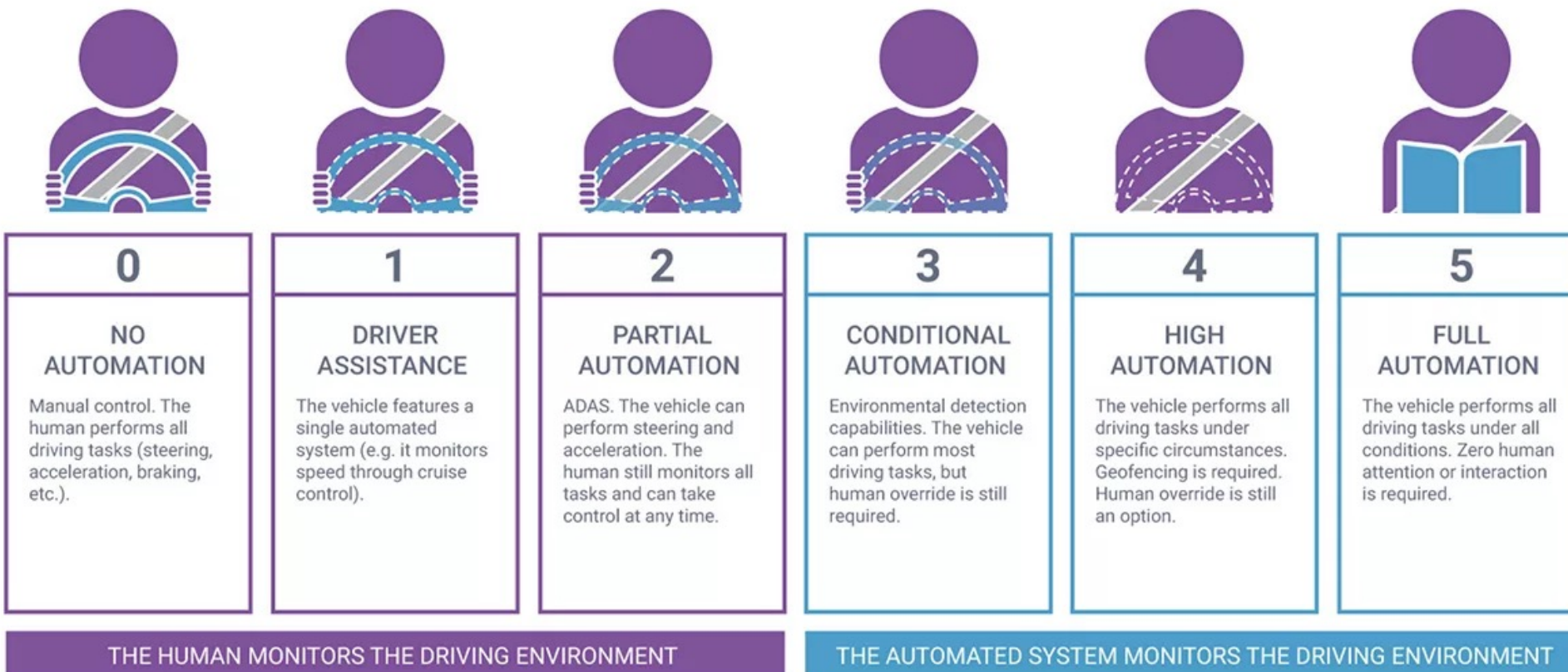
## ►► 构造优秀的研发助手

- 优秀的模型、推理框架与产品的协同让我们得以交付优秀的智能研发工具
- 基于大规模的落地使用，用户优先、反馈驱动使产品和模型不断改进，形成良性闭环
- 通过RAG强化、大模型思维链、平台开放等形式，智能研发将更贴近组织与团队，胜任更大规模的需求实现

# ▶▶ 人与机器的协同关系将迎来变革

**SYNOPSYS®**

## LEVELS OF DRIVING AUTOMATION



# 科技生态圈峰会 + 深度研习



—1000+ 技术团队的选择



 **K+峰会**  **上海站**

**K+ 全球软件研发行业创新峰会**

时间: 2024.06.21-22

 **K+峰会**  **敦煌站**

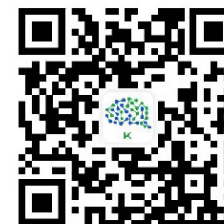
**K+ 思考周®研习社**

时间: 2024.10.17-19

 **K+峰会**  **香港站**

**K+ 思考周®研习社**

时间: 2024.11.10-12



K+峰会详情



 **AiDD峰会**  **上海站**

**AI+研发数字峰会**

时间: 2024.05.17-18

 **AiDD峰会**  **北京站**

**AI+研发数字峰会**

时间: 2024.08.16-17

 **AiDD峰会**  **深圳站**

**AI+研发数字峰会**

时间: 2024.11.08-09



AiDD峰会详情



# THANKS

