# 大模型使用指南——以ChatGPT为例

首先我们先来了解一下ChatGPT的大语言的缺点：

1. 无法引用信息来源。因为无法访问互联网，也无法记忆信息的来源，有时候能生成看起来不错的答案，但不准确；
2. 偏见和偏差。模型容易生成刻板的回答，也会说出有歧视性的回答；
3. 乱回答。有时候明明模型不知道正确的答案，但它依然会一本正经地胡说；
4. 不擅长数学。简单的数学解决起来就有困难，更复杂的数学就更难了；
5. 提示词骇客。就是指提问者有能力欺骗模型生成任何他们想要的答案。

所以，我们要摆正看待ChatGPT的态度，做到以下几点：

1. 不要把“思考"外包给ChatGPT，相反，要用ChatGPT提升我们的工作效率；
2. 不要用ChatGPT创造内容，相反，可以用它来搭建框架和做内容审阅;
3. 不要用ChatGPT生成引用文献，否则，它会生成一些瞎编乱造的内容;

那么我们怎么在日常交易工作中有效且高效使用GPT呢？这就需要理解GPT运行的内核：GPT就是对用户输入的话进行编码提取prompt（提示词）然后将概率最大的词或句子解码输出。因此，如何让模型能够准确地提取到提示词便是输出准确性的重要前提。要怎么才能准确呢？我们可以使用提问方法中最常用的"5W1H"方法，即"What, Who, Where, When, Why, andHow"，中文表述就是“是什么，是谁，在哪里，什么时候，为什么，怎么办"。从中我们能提炼出哪些信息来？首先是角色，你希望告诉ChatGPT他应该扮演个什么角色， 或者告诉他应该把我当做一个什么角色。其次是场景，什么时候，在哪里，这就是给ChatGPT提供更多的背景信息/上下文信息。然后是目的，当ChatGPT明白了你的目的之后，他的思考逻辑就会越符合你的要求。最后是方法，如果你还能告知ChatGPT应该按照什么样的方法去做件事，那么他给出的答案会更加符合你的要求。如果给不出，也没关系，可以让ChatGPT自己去想多种方法。下面分应用场景列出部门经常用到的指令

1. 代码生成
2. **为名为[名称]的[类/模块]生成具有以下功能的[语言]代码: [功能描述]**
3. 创建一个[语言]函数以使用以下输入对[数据结构]执行[操作]: [输入变量]和预期输出: [输出描述]
4. 为[领域]应用程序生成一个[语言]类，其中包括[方法列表]的方法和属性[属性列表]
5. 基于[设计模式]，用[语言]创建一个代码片段，演示其对[用例]的实现
6. 使用具有以下要求的[库/框架]编写[语言]脚本来执行[任务]: [要求列表]
7. 代码补全

1. 在[语言]中，完成以下用[值]初始化[数据结构]的代码片段: [代码]

1. 给定以下输入参数，完成计算[期望输出]的[语言]函数: [函数部分代码]
2. 完成[语言]代码以使用[参数]对[API端点]进行API调用并处理 响应: [代码]
3. 填写缺失的[语言]代码以实现以下功能的错误处理: [代码]
4. 完成以下遍历[数据结构]并执行[操作]的[语言]循环: [代码]
5. 错误检查
6. 确定以下[语言]代码片段中的任何潜在错误: [代码]
7. 分析给定的[语言]代码并提出改进建议以防止[错误类型]: [代码]
8. 在以下[语言]代码中查找任何内存泄漏并提出修复建议: [代码]
9. 检查给定[语言]代码中的任何竞争条件或并发问题: [代码]
10. 查看以下[语言]代码是否存在任何安全漏洞: [代码]
11. 代码审查
12. 查看以下[语言]代码以获得最佳实践并提出改进建议: [代码]
13. 分析给定的[语言]代码是否符合[编码风格指南] :[代码]
14. 检查以下[语言]代码以进行正确的错误处理并提出改进建议: [代码]
15. 评估给定[语言]代码的模块化和可维护性: [代码]
16. 评估以下[语言]代码的性能并提供优化建议: [代码]
17. 查询优化
18. 优化以下SQL查询以获得更好的性能: [query]
19. 分析给定的SQL查询是否存在任何潜在瓶颈: [query]
20. 为以下SQL查询建议索引|策略: [query]
21. 重写以下SQL查询以使用JOIN而不是子查询以提高性能: [query]
22. 为以下大规模数据库查询建议分区或分片策略: [SQL or NoSQL query]。
23. 代码重构
24. 建议对以下[语言]代码进行重构改进，以增强可读性和可维护性: [代码]
25. 确定在给定的[语言]代码中应用[设计模式]的机会: [代码]
26. 优化以下[语言]代码以获得更好的性能:[代码]
27. 重构给定的[语言]代码以提高其模块化和可重用性: [代码]
28. 建议更改给定的[语言]代码以遵守[编码风格或最佳实践]: [代码]