

19 | 如何实现波波熊学伴核心工作流（下）

月影 · 跟月影学前端智能体开发



你好，我是月影。

在上一节课，我们讲了生成内容大纲的部分，接下来我们继续深入，讨论如何实现内容的拆解以及文章的生成。


同样，我们首先回顾整体工作流：

目前我们实现到了撰写大纲，那么接下来，我们要根据大纲拆解内容了。

根据大纲拆解内容


首先，我们还是用 Trae 打开 Bearbobo Discovery 项目，创建新的提示词模板文件

`/lib/prompts/sub-topics.tpl.ts`，内容如下：

 复制代码

```
1 export default `
2 # Overall Rules to follow
3 1. Do response in 简体中文 and output **correct JSON Format ONLY**.
4 2. Do NOT explain your response.
5 3. DO NOT mention the student' Information when you generate the content.
6
7 ## Student Information
8 - gender: {{gender}}
9 - age: {{age}}
10 - student location: 中国
11
12 ## Study Style
13 The article must always adhere to the following elements:
14 - Communication-Style: Simple and Clear
15 - Tone-Style: Interesting and Vivid
16 - Reasoning-Framework: Intuitive
17 - Language: 简体中文
18
19 # Role and Goals
20 你正在模拟一个教育家，专门编写针对 {{age}} 岁学生的教学内容，采用<Communication-Style>的行式
21 1. You will receive an educational outline that includes 'topics'和'introduction'
22 2. [IMPORTANT!]该学生年龄是 {{age}} 岁，务必用适合学生年龄认知的问题来引导学生。
23
24 # Output Format(JSON)
25 你输出的 JSON 格式如下，这里有一个问题是“云是什么，我们能躺在云上吗？”的示例：
26 ```
27 {"topics":[{"topic":"云是由什么组成的，它们看起来是什么样的？","subtopics":["云主要由水蒸
28 ```
29 `;
```


这个提示词中，我们重点是大纲主题的拆解，其核心指令就是后面这句话：

 复制代码

```
1 You will receive an educational outline that includes 'topics'和'introduction',`
```

当 AI 收到大纲中的 topic 和 introduction 时，根据问题，将 topic 拆解成和问题直接相关的 subtopic（子主题）。

接下来我们修改 server.ts，改进 workflow：

 复制代码

```
1  ...
2  import subTopicsPrompt from './lib/prompts/sub-topics.tpl.ts';
3  ...
4
5  app.get('/generate', async (req, res) => {
6      const userConfig = {
7          gender: 'female',
8          age: '6',
9      };
10     const question = req.query.question as string;
11     const query = req.query.query as string;
12     let searchResults = '';
13     if (query) {
14         const queries = query.split(';');
15         const promises = queries.map((query) => search(query));
16
17         searchResults = JSON.stringify(await Promise.all(promises));
18     }
19     // ----- The work flow start -----
20     const ling = new Ling(config);
21     const quickAnswerBot = ling.createBot('quick-answer', {}, {
22         response_format: { type: 'text' }
23     });
24     quickAnswerBot.addPrompt(quickAnswerPrompt, userConfig);
25
26     const outlineBot = ling.createBot('outline');
27     outlineBot.addPrompt(outlinePrompt, userConfig);
28
29     outlineBot.addFilter('image_prompt');
30     outlineBot.addListener('string-response', ({ uri, delta }) => {
31         ling.handleTask(async () => {
32             if (uri.includes('image_prompt')) {
33                 // generate image
34                 const { url } = await generateImage(`A full-size picture suitable
35                 ling.sendEvent({ uri: 'cover_image', delta: url });
36             }
37         });
38     });
39
40     outlineBot.addListener('inference-done', (content) => {
```

```

41     const outline = JSON.parse(content);
42     delete outline.image_prompt;
43
44     const bot = ling.createBot();
45     bot.addPrompt(subTopicsPrompt, userConfig);
46     bot.addFilter(/\s/subtopics\s/);
47
48     bot.chat(JSON.stringify(outline));
49 });
50
51 if (searchResults) {
52     quickAnswerBot.addPrompt(`参考资料:\n${searchResults}`);
53     outlineBot.addPrompt(`参考资料:\n${searchResults}`);
54 }
55
56 quickAnswerBot.chat(question);
57 outlineBot.chat(question);
58
59 ling.close();
60
61 // setting below headers for Streaming the data
62 res.writeHead(200, {
63     'Content-Type': "text/event-stream",
64     'Cache-Control': "no-cache",
65     'Connection': "keep-alive"
66 });
67
68 pipeline((ling.stream as any), res);
69 });

```

对比一下上一节课的版本，其实 server 中最主要的改变是增加了当 outlineBot 推理完成之后的处理：

[📄 复制代码](#)

```
1 outlineBot.addListener('inference-done', (content) => {
2     const outline = JSON.parse(content);
3     delete outline.image_prompt;
4
5     const bot = ling.createBot();
6     bot.addPrompt(subTopicsPrompt, userConfig);
7     bot.addFilter(/\s/subtopics\s/);
8
9     bot.chat(JSON.stringify(outline));
10 });
```

在这里，我们等 outlineBot 完成推理后，将 content 取出，删掉不要的 image_prompt。因为这只是生成封面图的提示词，留着它输入给 subtopics 相关的处理，可能会影响 AI 的注意力，导致出来的结果不理想。

接着我们创建一个新的 bot，设置 subTopicsPrompt 提示词，最后将不用发送给前端的字段（主要是 subtopics 的部分）过滤掉，然后执行 bot.chat 即可。

现在我们修改一下前端 App.vue 代码：

[📄 复制代码](#)


```
1 <script setup lang="ts">
2 ...
3 import { set, get } from 'jsonuri';
4 ...
5 ...
6 const details: any = { topics: [] };
7 const topics: Ref<any[]> = ref([]);
8
9 const details: any = { topics: [] };
10
11 const topics: Ref<any[]> = ref([]);
12
13 const questionSelected = (question: string, index: number) => {
14     quickAnswer.value = '';
15     description.value = '';
16     const query = queries[index].join(';');
17     const endpoint = '/api/generate';
18     const eventSource = new EventSource(`${endpoint}?question=${question}&query=${q
19     eventSource.addEventListener("message", function (e: any) {
```

```

20     let { uri, delta } = JSON.parse(e.data);
21     if (uri.endsWith('quick-answer')) {
22         quickAnswer.value += delta;
23     }
24     if (uri.endsWith('introduction')) {
25         description.value += delta;
26     }
27     if (uri.endsWith('cover_image')) {
28         coverUrl.value = delta;
29     }
30     if (uri.startsWith('topics')) {
31         let content = get(details, uri) || '';
32         set(details, uri, content + delta);
33         topics.value = [...details.topics];
34     }
35 });
36 eventSource.addEventListener('finished', () => {
37     console.log('传输完成');
38     eventSource.close();
39 });
40 }
41 </script>
42
43 <template>
44 ...
45     <BookDetails :image="coverUrl" :expand="expand" :introduction="description" :
46 ...
47 </template>

```

最主要的部分是我们处理 topics 的逻辑分支：

 复制代码

```


1     if (uri.startsWith('topics')) {
2         let content = get(details, uri) || '';
3         set(details, uri, content + delta);
4         topics.value = [...details.topics];
5     }

```

在这里我们通过 jsonuri 的 get 和 set 方法来把接收到的增量数据更新到 details 对象中，再通过 details 对象更新 topics 数据，从而更新 UI。

注意这里一个小细节，我们要提前定义好数据类型：`const details: any = { topics: [] }`；让 details 变量的 topics 是一个数组，这样 jsonuri 更新的 topics 数据才会转成数组，否则的话，它会被转成 key 为"0"“1”、“2"的对象。

我们有了 topics 数据，还要把它传给 BookDetails 组件进行渲染，所以我们要修改 BookDetails.vue：

 复制代码


```
1 <script setup lang="ts">
2 import { marked } from 'marked';
3 import { type PropType } from 'vue';
4
5 interface Topic {
6   topic: string,
7   post_reading_question: string,
8   article?: string,
9   image_prompt?: string,
10 }
11
12 defineProps({
13   image: {
14     type: String,
15     default: '',
16   },
17   question: {
18     type: String,
19     default: '',
20   },
21   introduction: {
22     type: String,
23     default: '',
24   },
25   expand: {
26     type: Boolean,
27     default: false,
28   },
29   topics: {
30     type: Array as PropType<Topic[]>,
31     default: [],
32   }
33 });
34 </script>
35
36 <template>
37   <div v-if="expand" class="details" @click.stop="">
```

```
38     <div class="cover">
39         
41     </div>
42     <h1>{{ question }}</h1>
43     <p class="introduction" v-html="marked.parse(introduction)"></p>
44     <div class="article">
45         <div v-for="topic in topics" class="topic">
46             <h3 class="topic-title">{{ topic.topic }}</h3>
47             <p class="post-reading-question">{{ topic.post_reading_question }}
48         </div>
49     </div>
50 </div>
51 </template>
52
53 <style scoped>
54 .details {
55     position: absolute;
56     top: 90px;
57     width: 600px;
58     height: 800px;
59     overflow-y: auto;
60     background-color: white;
61     box-shadow: #aaa 0px 0px 10px 10px;
62 }
63
64 .cover {
65     height: 260px;
66     overflow: hidden;
67     position: relative;
68 }
69
70 .img-fluid {
71     position: absolute;
72     left: 50%;
73     top: 50%;
74     transform: translate(-50%, -50%);
75     width: 100%;
76 }
77
78 .introduction {
79     padding: 20px;
80     font-size: 1rem;
81     border-bottom: solid 1px #ccc;
82 }
83
84 .article {
85     padding: 20px;
86     text-align: start;
```



```
87 }  
88 </style>
```

我们增加了 topics 属性，它是一个 Topic 数组，现在我们只获得了 topic 和 post_reading_question 两部分数据，我们先把它们展示出来。

 复制代码

```
1 <div class="article">  
2   <div v-for="topic in topics" class="topic">  
3     <h3 class="topic-title">{{ topic.topic }}</h3>  
4     <p class="post-reading-question">{{ topic.post_reading_question }}</p>  
5   </div>  
6 </div>
```


最终代码运行效果如下图：

这样我们就实现了大纲拆解部分，接下来就是最后的文章生成了。

根据拆解后的大纲生成正文

接下来我们生成文章。同样地，我们先添加提示词模块：

```
/lib/prompts/article.tpl.ts
```

 复制代码

```
1 export default `# Overall Rules to follow  
2 1. Do response in 简体中文 and output **correct JSON Format ONLY**.  
3 2. Do NOT explain your response.
```

```

4 3. DO NOT mention the student' Information when you generate the content.
5
6 ## Student Information:
7 - gender: {{gender}}
8 - age: {{age}}
9 - student location: 中国
10
11 ## Study Style
12 The students' learning style preferences
13 - Communication-Style: Simple and Clear
14 - Tone-Style: Interesting and Vivid
15 - Reasoning-Framework: Intuitive
16 - Language: 简体中文
17
18 # Role and Goals
19 你是波波熊，你正在和其它作家共同编写一个文章，你的任务为 {{age}} 岁，处于 {{config.location}}
20 1. You will receive a JSON that includes 'topic','subtopics','post_reading_question'
21 2. 首先，明确该部分的主要主题(topic)和知识点(subtopics)，这有助于文章结构的清晰性，让学生们能
22 3. 【重要!】不要在开场打招呼，避免使用<AvoidKeywords>中的任何词语。
23 4. 明确目标读者的年龄和知识水平，该学生年龄是 {{age}} 岁，所以你采用的语言和内容要贴近学生能接受
24 5. 使用<Communication-Style>的风格和<Tone-Style>的写作风格。如果学生年龄较小，描述云时使用
25 6. 将内容分成小段，自然的串联起知识点，使用段落叙述。这有助于学生逐步理解，不至于感到信息过载，确
26 7. 使用具有**极具画面感**的语言编写，把你写好的文章段落存储于'article_paragraph'内。
27 {% if (age > 7) %}
28     【重要!】如果是数学、自然科学和科普类主题，介绍概念的同时，尽量深入解释原理。
29 {% endif %}
30 8. 务必确保文章内容回答了'post_reading_question'中的问题。
31 9. 根据 topic 定制一个图像提示，存储于'image_prompt'。
32 10. 生成具有古典风格 'image_title' 和一个打油诗风格的'poetic_line'。
33
34 ## AvoidKeywords
35 ['魔法', '超级英雄', '想象一下', '你知道吗? ']
36
37 # Output Format(JSON)
38 你输出的 JSON 格式如下，这里有一个主题是“云是由什么组成的，它们看起来是什么样的？”的示例：
39 \\\`
40 {"article_paragraph":"云是由什么组成的呢？主要成分是水蒸气，一种无色无味的气体，悄无声息地弥漫
41 \\\`
42 `;

```


在这个提示词中，我们以大纲拆解后的内容作为输入，分别创建每个章节，输出 article_paragraph、image_prompt，另外还有 image_title 和 poetic_line 是用来保留作为答题奖励卡片的。不过我们在课程里把这些不重要的部分简化掉了，暂时不使用这两个字段。

接着我们继续改写 server.ts:

```
1 ...
2 import articleTpl from './lib/prompts/article.tpl.ts';
3 ...
4
5 app.get('/generate', async (req, res) => {
6   const userConfig = {
7     gender: 'female',
8     age: '6',
9   };
10  const question = req.query.question as string;
11  const query = req.query.query as string;
12  let searchResults = '';
13  if (query) {
14    const queries = query.split(';');
15    const promises = queries.map((query) => search(query));
16
17    searchResults = JSON.stringify(await Promise.all(promises));
18  }
19  // ----- The work flow start -----
20  const ling = new Ling(config);
21  const quickAnswerBot = ling.createBot('quick-answer', {
22    max_tokens: 4096 * 4,
23  }, {
24    response_format: { type: 'text' }
25  });
26  quickAnswerBot.addPrompt(quickAnswerPrompt, userConfig);
27
28  const outlineBot = ling.createBot('outline');
29  outlineBot.addPrompt(outlinePrompt, userConfig);
30
31  outlineBot.addFilter('image_prompt');
32  outlineBot.addListener('string-response', ({ uri, delta }) => {
33    ling.handleTask(async () => {
34      if (uri.includes('image_prompt')) {
35        // generate image
36        const { url } = await generateImage(`A full-size picture suitable
37        ling.sendEvent({ uri: 'cover_image', delta: url });
38      }
39    });
40  });
41
42  outlineBot.addListener('inference-done', (content) => {
43    const outline = JSON.parse(content);
44    delete outline.image_prompt;
45
46    const bot = ling.createBot();
47    bot.addPrompt(subTopicsPrompt, userConfig);
```

```
48     bot.addFilter(/\//subtopics\//);
49
50     bot.chat(JSON.stringify(outline));
51
52     // 文章生成
53     bot.addListener('inference-done', (content) => {
54         const { topics } = JSON.parse(content);
55         for (let i = 0; i < topics.length; i++) {
56             const topic = topics[i];
57
58             const bot = ling.createBot(`topics/${i}`);
59             bot.addPrompt(articleTpl, userConfig);
60             bot.addFilter({
61                 article_paragraph: true,
62                 image_prompt: true,
63             });
64
65             bot.addListener('inference-done', (content) => {
66                 console.log(JSON.parse(content));
67             });
68             bot.chat(JSON.stringify(topic));
69         }
70     });
71 });
72
73 if (searchResults) {
74     quickAnswerBot.addPrompt(`参考资料:\n${searchResults}`);
75     outlineBot.addPrompt(`参考资料:\n${searchResults}`);
76 }
77
78 quickAnswerBot.chat(question);
79 outlineBot.chat(question);
80
81 ling.close();
82
83 // setting below headers for Streaming the data
84 res.writeHead(200, {
85     'Content-Type': "text/event-stream",
86     'Cache-Control': "no-cache",
87     'Connection': "keep-alive"
88 });
89
90 pipeline((ling.stream as any), res);
91 });
```

最核心的逻辑就是，我们在 subtopics 生成后，对应 bot 的 `inference-done` 事件中，循环遍历每个章节，用章节的 subtopics 内容去并行生成文章。


 复制代码

```
1 for (let i = 0; i < topics.length; i++) {
2   const topic = topics[i];
3
4   const bot = ling.createBot(`topics/${i}`);
5   bot.addPrompt(articleTpl, userConfig);
6   bot.addFilter({
7     article_paragraph: true,
8     image_prompt: true,
9   });
10
11   bot.addListener('inference-done', (content) => {
12     console.log(JSON.parse(content));
13   });
14   bot.chat(JSON.stringify(topic));
15 }
16
```

注意一个细节，我们前端只需要用到 `article_paragraph` 和 `image_prompt` (`image_prompt` 留待下一个章节介绍生成插画的时候使用)，所以我们可以添加过滤器，传一个对象 Map 可以告诉 Ling 我们只需要这些字段的内容。

这样我们就完成了服务端的部分，因为我们直接将文章 bot 的输出 root 指定为 `topics/${i}`，所以它的内容会输出到 `details.topics` 对象上去，这样我们就不用修改 `App.vue` 了。

前端我们只需要修改 `BootDetails` 组件，添加 `article_paragraph` 的展示就可以了。

 复制代码

```
1 ...
2 <div class="article">
3   <div v-for="topic in topics" class="topic">
4     <h3 class="topic-title">{{ topic.topic }}</h3>
5     <p class="article_paragraph" v-html="marked.parse(topic.article_paragraph)">{{ topic.article_paragraph }}</p>
6     <p class="post-reading-question">{{ topic.post_reading_question }}</p>
7   </div>
```

```
8     </div>
9     ...
```

我们现在运行代码，得到效果如下：

要点总结

至此，我们把波波熊学伴的主体流程都讲完了，它是波波熊学伴产品最核心的部分。

开发 AI 应用实际上最核心的就是开发大模型的工作流，需要梳理输入输出，也要定义每个节点的输入输出数据格式和结构。

有了 Ling 框架的帮助，我们能够非常快速和方便地处理每个流程节点的数据，并且能够第一时间将更新的数据发送给前端，所以能够达到比较好的效果。

课后练习

因为时间所限，我只分析了主体流程，至于很多技术细节，实际上在实战代码中还有许多值得研究和思考的部分。你可以课后试着分析一下 server 代码，说说看为什么波波熊学伴生成复杂结构内容的响应速度能够这么快，在这些细节里还有没有进一步优化的空间，欢迎大家把想法发到评论区讨论。

波波熊学伴的完整源代码详见 [🔗 GitHub 仓库](#)。

欢迎你在留言区和我交流互动，如果这节课对你有启发，别忘了分享给身边更多朋友。

AI智能总结

Bearbobo Discovery项目的核心 workflow 包括内容拆解和文章生成。项目使用Trae打开，创建新的提示词模板文件`/lib/prompts/sub-topics.tpl.ts`。文章必须遵循特定的沟通风格、语气和结构化思维方式。教育家需要根据提供的大纲将主题分解为与问题相关的子主题，避免重复知识点，并重点关注适合学生年龄认知的问题来引导学生。

在服务器端的工作流中，当outlineBot完成推理后，将content取出，删掉不需要的image_prompt，并创建一个新的bot，设置subTopicsPrompt提示词，最后将不用发送给前端的字段过滤掉，然后执行bot.chat。前端代码中，通过处理topics的逻辑分支，将接收到的增量数据更新到details对象中，再通过details对象更新topics数据，从而更新UI。最后，将topics数据传给BookDetails组件进行渲染。

接下来，根据拆解后的大纲生成正文。首先添加提示词模块`/lib/prompts/article.tpl.ts`，要求围绕上次的总结结果和上次总结后续的文章找出文章的重点，过滤掉非话题的重点和和知识无关的重点，从中筛选返回不超过10个的重点，重点简洁清晰不要太长，返回内容不超过2000字。

这些工作流程和代码修改将有助于实现Bearbobo Discovery项目中的大纲拆解和文章生成的功能。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。