AI 智能·学习搭子

技术架构文档

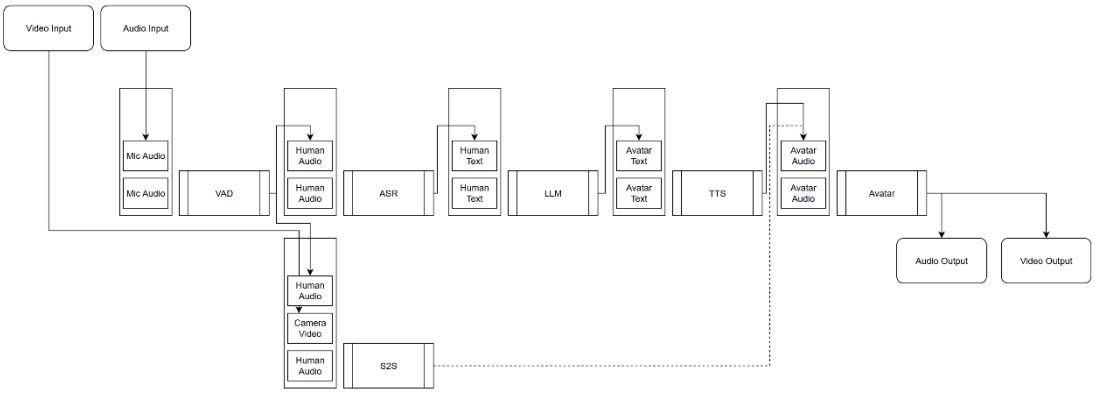
发包方：数字马力

承接方：声像科技

日期：2025 年 10 月

数字人伴学系统

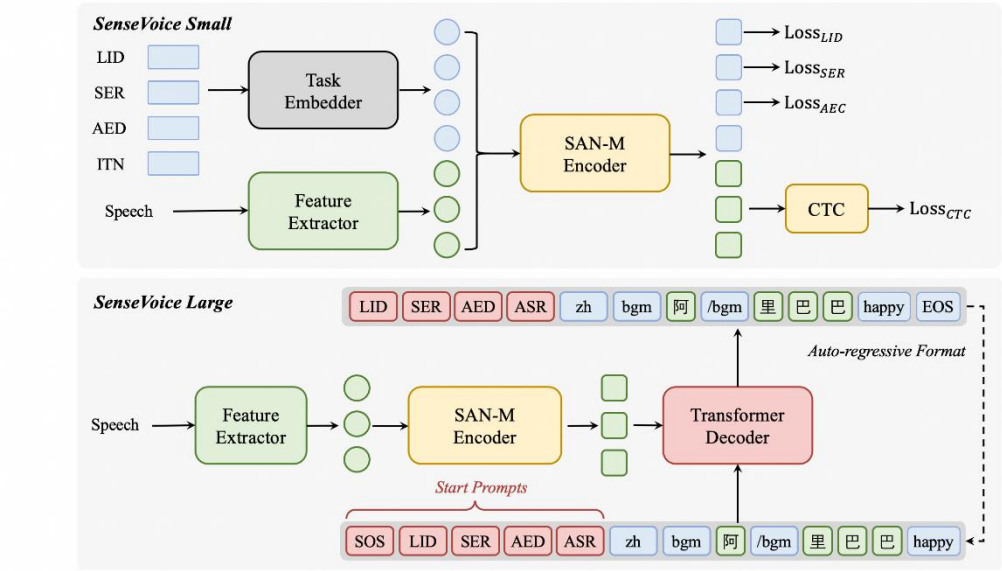
本系统以阿里巴巴通义实验室开源的 OpenAvatarChat 为基座，构建了一套端到端的实时数字人对话工作流。该工作流采用模块化设计，将语音识别（ASR）、大语言模型（LLM）、语音合成（TTS）和数字人驱动（Avatar）四大核心引擎解耦，便于独立优化与替换。



在本地部署测试环境中（Ultra7-265K CPU + Nvidia RTX 5080 GPU），我们选用了以下技术栈，并在局域网 HTTPS（由 OpenSSL 提供 SSL 证书）环境下进行推流，实现了约 3 秒的平均端到端延迟。以下是对四大功能引擎的选型介绍与优化方案：

ASR 语音识别引擎：SenseVoice

SenseVoice 是阿里达摩院推出的工业级语音识别基础模型，不仅支持高精度的自动语音识别（ASR），还集成了语种识别（LID）、情感识别（SER）等多模态能力，能为后续的个性化交互提供丰富的上下文信息。



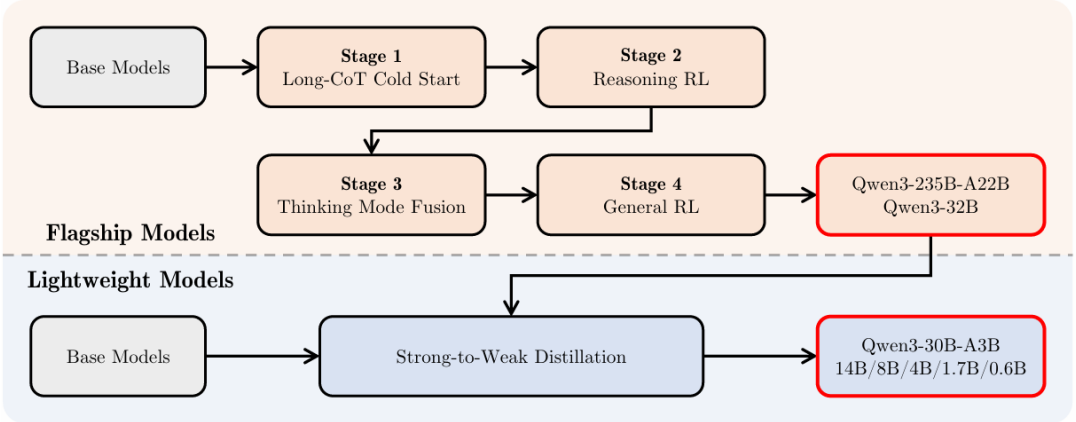
为降低交互延迟，我们对 SenseVoice 进行了伪流式处理改造。通过将用户的连续语音流按固定时间窗口（chunk）进行分块，并采用截断注意力机制（truncated attention）对每个语音块进行独立推理。这种方式在保证识别准确率的同时，显著减少了用户说完话后到系统开始响应的等待时间，提升了对话的流畅感。

核心代码示例如下：

```python
class StreamingSenseVoice:
    def inference(self, speech):
        speech = speech[None, :, :]
        speech_lengths = torch.tensor([speech.shape[1]])
        speech = speech.to(self.device)
        speech_lengths = speech_lengths.to(self.device)
        speech = torch.cat((self.query, speech), dim=1)
        speech_lengths += 4
        encoder_out, _ = self.model.encoder(speech, speech_lengths)
        return self.model.ctc.log_softmax(encoder_out)[0, 4:]

    def decode(self, times, tokens):
        times_ms = []
        for step, token in zip(times, tokens):
            if len(self.tokenizer.decode(token).strip()) == 0:
                continue
            times_ms.append(step * 60)
        return times_ms, self.tokenizer.decode(tokens)

    def streaming_inference(self, audio, is_last):
        self.fbank.accept_waveform(audio, is_last)
        features = self.fbank.get_lfr_frames(
            neg_mean=self.neg_mean, inv_stddev=self.inv_stddev
        )
        if is_last and len(features) == 0:
            features = self.zeros
        for idx, feature in enumerate(torch.unbind(torch.tensor(features), dim=0)):
            is_last = is_last and idx == features.shape[0] - 1
            self.caches = torch.roll(self.caches, -1, dims=0)
            self.caches[-1, :] = feature
            self.cur_idx += 1
            cur_size = self.get_size()
            if cur_size != self.chunk_size and not is_last:
                continue
            probs = self.inference(self.caches)[self.padding :]
            if cur_size != self.chunk_size:
                probs = probs[self.chunk_size - cur_size :]
            if not is_last:
                probs = probs[: self.chunk_size]
            if self.beam_size > 1:
                res = self.decoder.ctc_prefix_beam_search(
                    probs, beam_size=self.beam_size, is_last=is_last
                )
                times_ms, text = self.decode(res["times"][0], res["tokens"][0])
            else:
                res = self.decoder.ctc_greedy_search(probs, is_last=is_last)
                times_ms, text = self.decode(res["times"], res["tokens"])
            yield {"timestamps": times_ms, "text": text}
```

LLM 大语言模型引擎：Qwen3-4B + AgentUniverse

Qwen3-4B 是阿里巴巴最新一代的大语言模型，以其卓越的参数效率和强大的多语言、工具调用能力著称。其性能可与上一代更大参数量的模型相媲美，非常适合在本地有限算力下部署。



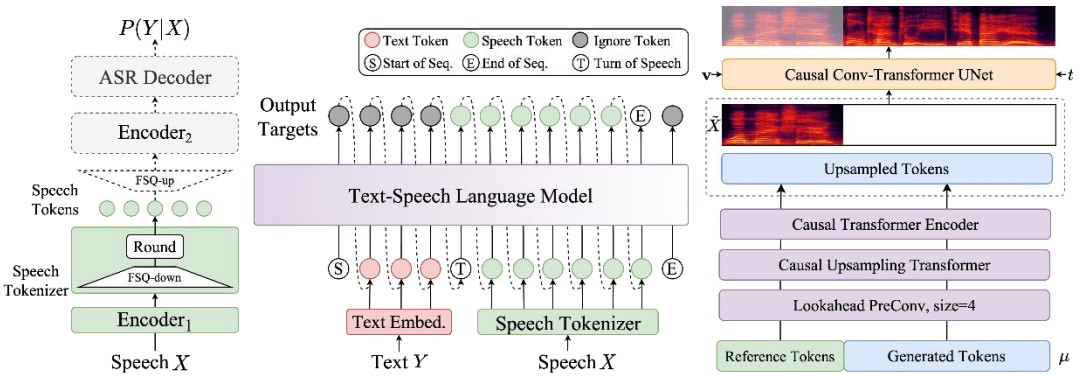
我们将 Qwen3-4B 深度集成到 AgentUniverse 多智能体框架中。数字人不再是一个孤立的对话模型，而是作为一个“伴学智能体”（Companion Agent），能够与其他智能体（如“知识库检索智能体”、“学习规划智能体”）协同工作。例如，当学生提出一个复杂问题时，伴学智能体会调用知识库智能体检索最新资料，并将结果整合后以更易懂的方式回答，实现了知识的动态更新与深度整合。

核心代码示例如下：

```python
class QWenOpenAIStyleLLM(OpenAIStyleLLM):
    api_key: Optional[str] = Field(default_factory=lambda:
get_from_env("DASHSCOPE_API_KEY"))
    api_base: Optional[str] = Field(default_factory=lambda: get_from_env(
        "DASHSCOPE_API_BASE") or "https://dashscope.aliyuncs.com/compatible-mode/v1")
    proxy: Optional[str] = Field(default_factory=lambda: get_from_env("DASHSCOPE_PROXY"))
    organization: Optional[str] = Field(default_factory=lambda:
get_from_env("DASHSCOPE_ORGANIZATION"))

    def _call(self, messages: list, **kwargs: Any) -> Union[LLMOutput,
Iterator[LLMOutput]]:
        return super()._call(messages, **kwargs)

    async def _acall(self, messages: list, **kwargs: Any) -> Union[LLMOutput,
AsyncIterator[LLMOutput]]:
        return await super()._acall(messages, **kwargs)

    def max_context_length(self) -> int:
        if super().max_context_length():
            return super().max_context_length()
        return QWen_Max_CONTEXT_LENGTH.get(self.model_name, 8000)

    def get_num_tokens(self, text: str) -> int:
        tokenizer = get_tokenizer(self.model_name)
        return len(tokenizer.encode(text))
```

TTS 语音合成引擎：CosyVoice 2.0

CosyVoice 2.0 在发音准确性、语音自然度和响应速度上均有显著提升，其发音错误率相比 1.0 版本降低了 30%-50%。更重要的是，它支持通过自然语言指令控制语音的情感和韵律，这对于营造有温度的伴学体验至关重要。



(a) Supervised Speech Tokenizer (b) Text-Speech Language Model (c) Causal Flow Matching

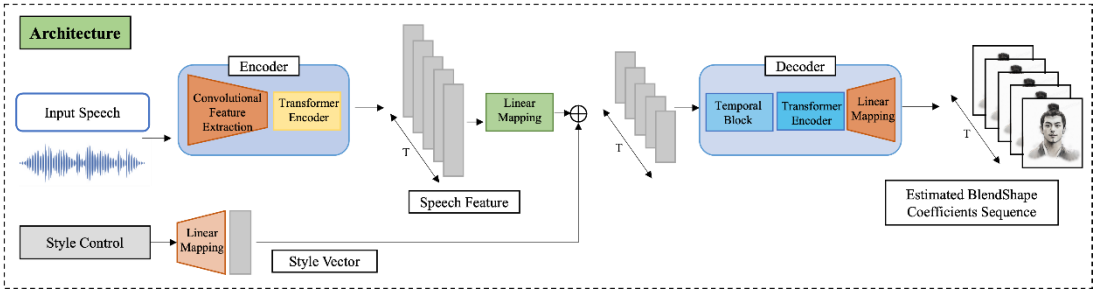
我们利用其情感控制能力，根据 LLM 对对话上下文的情感分析结果，动态调整 TTS 输出的语调。例如，在学生回答正确时，数字人会用更欢快、鼓励的语气；在学生遇到困难时，则会切换到更温和、耐心的语气，实现真正的情感化陪伴。

核心代码示例如下：

```python
class HandlerTTS(HandlerBase, ABC):
    def start_context(self, session_context, context: HandlerContext):
        context = cast(TTSContext, context)
        output_definition = self.get_handler_detail(session_context, context).outputs.get(ChatDataType.AVATAR_AUDIO).definition

        def task_consumer(task_inner_queue: deque, callback: callable):
            while True:
                if len(task_inner_queue) == 0:
                    time.sleep(0.03)
                    continue
                task = task_inner_queue[0]
                task = cast(HandlerTask, task)
                if task is None:
                    break
                try:
                    audio = task.result_queue.get(timeout=1)
                    if audio is not None:
                        output = DataBundle(output_definition)
                        output.set_main_data(audio)
                        output.add_meta("avatar_speech_end",
                         False if not task.speech_end else True)
                        output.add_meta("speech_id", task.speech_id)
                        callback(output)
                        if context.dump_audio:
                            dump_audio = audio
                            context.audio_dump_file.write(dump_audio.tobytes())
                    else:
                        task_inner_queue.popleft()
                except Exception as e:
                    logger.debug(e)

        context.task_consume_thread = threading.Thread(target=task_consumer, args=[context.task_queue, context.submit_data])
        context.task_consume_thread.start()
        self.task_queue_map[context.session_id] = context.task_queue
```

Avatar 数字人驱动引擎：LAM

LAM 是一种创新的大型数字人模型，能够从单张图片快速生成超写实的 3D 高斯头像，并支持跨平台实时驱动与渲染。这极大地简化了数字人形象的创建流程，未来可支持学生上传自己的照片生成专属学习伙伴。



核心代码示例如下：

```python
class HandlerAvatarLAM(HandlerBase):
    def handle(self, context: HandlerContext, inputs: ChatData,
               output_definitions: Dict[ChatDataType, HandlerDataInfo]):
        ...
        audio = inputs.data.get_main_data()
        audio_segments = queue.Queue()
        for audio_segment in slice_data(context.input_slice_context, audio.squeeze()):
            audio_segments.put_nowait(audio_segment)
        if speech_end:
            end_segment = context.input_slice_context.flush()
            if end_segment is not None:
                audio_segments.put_nowait(end_segment)
        if audio_segments.empty() and speech_end:
            audio_segments.put_nowait(np.zeros([50], dtype=np.float32))
        while not audio_segments.empty():
            t_start = time.monotonic()
            audio_segment = audio_segments.get_nowait()
            result, context_update = self.infer.infer_streaming_audio(
                audio=audio_segment,
                ssr=context.config.audio_sample_rate,
                context=context.inference_context,
            )
            context.inference_context = context_update
            need_flush = speech_end and audio_segments.empty()
            output = DataBundle(output_definition)
            arkit_data = result.get("expression")
            start_of_stream = speech_id != context.last_speech_id
            output.set_main_data(arkit_data.astype(np.float32))
            output.set_data("avatar_audio", audio_segment[np.newaxis, ...])
            output.add_meta("speech_id", speech_id)
            output.add_meta("avatar_speech_end", need_flush)
            output.start_of_stream = start_of_stream
            output.end_of_stream = need_flush
            if speech_text is not None:
                output.add_meta("avatar_speech_text", speech_text)
            dur_inference = time.monotonic() - t_start
            context.submit_data(output)
            context.last_speech_id = speech_id
```

与蚂蚁集团生态的融合

Qwen 大模型：智能内核，源于阿里、服务教育

Qwen 系列大模型由阿里巴巴通义实验室研发，是蚂蚁集团 AI 技术生态的重要组成部分。在数字人伴学系统中，我们采用 Qwen3-4B 作为核心语言理解与生成引擎，其融合体现在：

能力先进性：Qwen3 在参数效率、多语言支持、工具调用（Function Calling）等方面表现卓越，能在有限本地算力下提供接近更大模型的对话质量，契合高校学生对知识广度与深度的需求。

与 AgentUniverse 协同：Qwen 作为“伴学智能体”的大脑，深度集成于蚂蚁集团开源的 AgentUniverse 多智能体框架中，可主动调用知识库检索、学习规划等其他智能体，实现从“被动问答”到“主动引导”的跃迁。

安全可控：依托蚂蚁集团在大模型内容安全、价值观对齐方面的成熟机制，确保数字人输出内容符合教育伦理，杜绝有害或误导性信息。

LAM 数字人驱动引擎：超写实形象，快速生成，沉浸交互

LAM（Large Avatar Model）是通义实验室推出的大型数字人模型，代表了蚂蚁-阿里生态在 AIGC 与虚拟人领域的前沿探索。其融合价值在于：

形象个性化：LAM 支持从单张图片快速生成高保真 3D 数字人形象，未来可让学生上传照片生成“专属学习搭子”，极大提升情感连接与使用黏性。

实时驱动与渲染：LAM 具备跨平台实时驱动能力，结合情感识别结果，可实现口型、表情、眼神的自然同步，营造沉浸式陪伴感。

轻量化部署：LAM 优化了推理效率，可在配备消费级 GPU 乃至手机等移动设备端上流畅运行，降低高校或学生个人的使用门槛。

CosyVoice 2.0：情感化语音合成，让声音有温度

CosyVoice 是通义实验室推出的语音大模型，其 2.0 版本在自然度、可控性和多语言支持上显著提升。在系统中的融合体现为：

情感韵律控制：通过自然语言指令（如“用鼓励的语气”、“用耐心的语调”），动态调节语音的情感色彩，使数字人不仅是“会说话”，更是“懂情绪”的学习伙伴。

低错误率与高自然度：发音错误率较 1.0 版本降低 30%-50%，确保专业术语（如化学式、数学符号）的准确朗读，提升学习体验的专业性。

与 Qwen 联动：LLM 在生成文本时即可预置情感标签，TTS 引擎据此自动匹配语音风格，实现端到端的情感化语音输出。

SenseVoice：多模态语音识别，理解更全面

SenseVoice 是达摩院推出的工业级语音基础模型，集 ASR、语种识别（LID）、情感识别（SER）于一体。其融合优势在于：

伪流式处理优化：团队采用分块+截断注意力机制，在保证识别准确率的同时，将语音输入到系统响应的延迟压缩至可接受范围（约 1 秒），提升对话流畅性。

情感与语境感知：不仅识别“说了什么”，还能感知“以什么情绪说的”。该情感信息可同步至 LAM（驱动表情）和 CosyVoice（调整回应语气），形成“听-感-说-演”的多模态闭环。

多语种支持：天然支持中、英、日等多语种识别，为英语、日语等语言学科的游戏化学习平台提供底层语音交互能力。

整体生态协同：安全、规范、可扩展

除上述技术组件外，数字人伴学系统在整体架构上全面融入蚂蚁集团技术生态：

安全合规：严格遵循企业文档要求，采用 OAuth2 + JWT 进行身份认证，HTTPS（OpenSSL）加密通信，防范 XSS/CSRF 等 Web 攻击，保障学生隐私数据安全。

UI/UX 统一：前端界面将使用基于 Ant Design 的组件库进行开发，确保与蚂蚁系产品体验一致，降低用户学习成本。

协议标准化：系统设计遵循 MCP/A2A 智能体协议规范，未来可作为标准服务，无缝接入蚂蚁教育生态或其他第三方应用，实现能力开放与生态共建。

团队就数字人伴学系统申请了两项软件著作权并取得授权：

1. 软件名称：《面向深度伪造音视频的智能检测软件》

   证书号：软著登字第 16024611 号

   授权日期：2025 年 7 月 25 日

2. 软件名称：《终身学伴-数字虚拟人合成平台》

   证书号：软著登字第 16545571 号

   授权日期：2025 年 9 月 28 日


中华人民共和国国家版权局
计算机软件著作权登记证书

证书号：软著登字第16024611号

软件名称：面向深度伪造音视频的智能检测软件
[简称：音盾视卫]
V1.0

著作权人：吴育金;丁炜宸;王宇涵;[redacted]

权利取得方式：原始取得

权利范围：全部权利

登记号：2025SR1368413

根据《计算机软件保护条例》和《计算机软件著作权登记办法》的规定，经中国版权保护中心审核，对以上事项予以登记。

2025年07月25日


中华人民共和国国家版权局
计算机软件著作权登记证书

证书号：软著登字第16545571号

软件名称：终身学伴-数字虚拟人合成平台
[简称：终身学伴]
V1.0

著作权人：丁炜宸

权利取得方式：原始取得

权利范围：全部权利

登记号：2025SR1889373

根据《计算机软件保护条例》和《计算机软件著作权登记办法》的规定，经中国版权保护中心审核，对以上事项予以登记。

2025年09月28日

多智能体-知识库协作系统

本系统是“AI 智能 • 学习搭子”解决方案的知识中枢与智能引擎。它并非一个孤立的问答机器人，而是依托蚂蚁集团在复杂金融业务场景中淬炼出的 AgentUniverse 多智能体框架，构建了一个由多个领域专家智能体协同工作的动态知识网络。其核心价值在于，能够将静态的知识库转化为可推理、可协作、可进化的活知识，为数字人伴学系统和评估系统提供强大、精准、实时的智能支持。

AgentUniverse 是一个面向复杂业务场景设计的多智能体协作框架，其核心是一个“模式工厂（pattern factory）”，允许开发者对多智能体协作模式进行开发和定制。我们基于此框架，设计了以下关键智能体及其协作流程：

知识库检索智能体（Knowledge Retrieval Agent）：

作为系统的“图书管理员”，负责监听来自数字人伴学系统或评估系统的查询请求。它能理解查询的语义，并在我们构建的多学科、多来源（教材、论文、开源项目、行业报告）的知识库中进行高效、精准的检索。核心代码示例如下：

```python
class Knowledge(ComponentBase):
    def update_knowledge(self, **kwargs) -> None:
        document_list: List[Document] = self._load_data(**kwargs)
        document_list = self._update_process(document_list)
        futures = []
        if "stores" in kwargs:
            stores = kwargs["stores"]
        else:
            stores = self.stores
        for _store_code in stores:
            futures.append(
                self.insert_executor.submit(
                    StoreManager().get_instance_obj(_store_code).update_document,
                    document_list))
        wait(futures, return_when=ALL_COMPLETED)
        for future in futures:
            future.result()
```

内容生成与摘要智能体（Content Generation & Summarization Agent）：

作为系统的“内容编辑”，接收检索智能体返回的原始资料。它利用大语言模型（如 Qwen3）的能力，对冗长、专业的原始信息进行提炼、总结、改写，生成适合学生当前认知水平和学习阶段的、易于理解的答案或学习材料。

```
[CORE_PACKAGE]
default = ['discussion_group_app.intelligence.agentic']
agent = ['discussion_group_app.intelligence.agentic.agent']
knowledge = ['discussion_group_app.intelligence.agentic.knowledge']
llm = ['discussion_group_app.intelligence.agentic.llm']
planner = []
tool = ['discussion_group_app.intelligence.agentic.tool']
memory = ['discussion_group_app.intelligence.agentic.memory']
service = ['discussion_group_app.intelligence.service.agent_service']
prompt = ['discussion_group_app.intelligence.agentic.prompt']
store = ['discussion_group_app.intelligence.agentic.knowledge.store']
rag_router = ['discussion_group_app.intelligence.agentic.knowledge.rag_router']
doc_processor = ['discussion_group_app.intelligence.agentic.knowledge.doc_processor']
query_paraphraser =
['discussion_group_app.intelligence.agentic.knowledge.query_paraphraser']
memory_compressor = ['discussion_group_app.intelligence.agentic.memory.memory_compressor']

memory_storage = ['discussion_group_app.intelligence.agentic.memory.memory_storage']
```

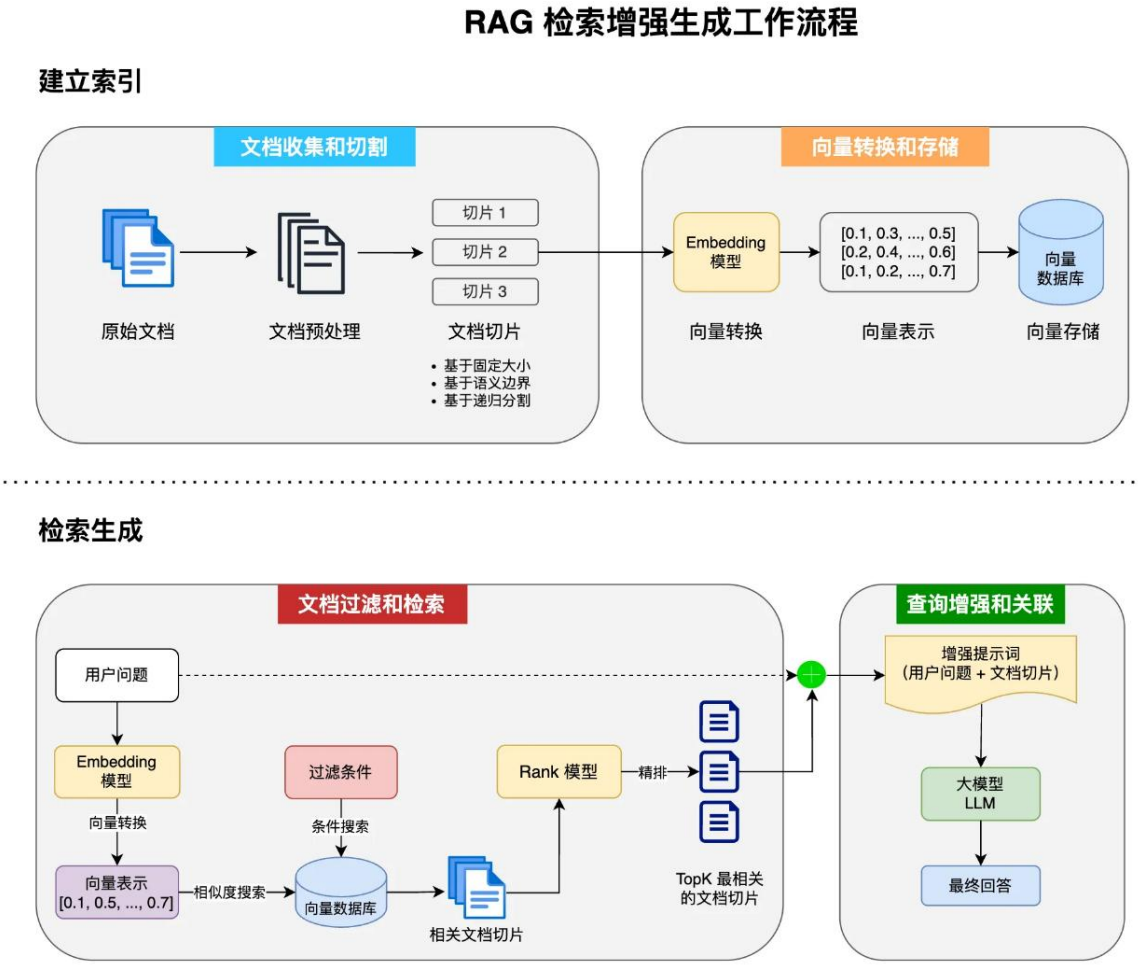
学习规划智能体（Learning Planner Agent）：

作为系统的“学业顾问”，它与“多学科客制化评估系统”深度联动。当评估系统诊断出学生的知识薄弱点后，该智能体会综合学生的学习目标（校园考试/就业技能）、历史学习数据和知识库内容，动态生成或调整个性化的学习路径和任务清单。

```
introduction: 你是一位专业的学业规划顾问，精通教育心理学和学习科学。
target: 基于多学科客制化评估系统的诊断结果，为学生制定个性化的学习路径和任务清单，帮助学生实现学习目标。
instruction: |
  你需要遵守的规则是：
  1. 深度分析评估系统提供的学生知识薄弱点诊断结果，准确理解学生的学习现状。
  2. 综合考虑学生的学习目标类型（校园考试提升/就业技能培养），制定针对性的学习策略。
  3. 基于学生的历史学习数据，识别学习偏好、学习节奏和有效的学习方式。
  4. 利用知识库内容，为每个薄弱知识点匹配最适合的学习资源和练习材料。
  5. 动态生成分阶段的学习路径，包括：基础巩固→能力提升→综合应用三个层次。
  6. 制定具体可执行的任务清单，包括学习时间安排、里程碑检查点和评估标准。
  7. 根据学生的学习进度反馈，实时调整和优化学习计划。
  8. 提供学习方法指导，包括记忆技巧、理解策略和应用练习建议。
  9. 设置激励机制和成就感培养，保持学生的学习动力。
  10. 与评估系统保持联动，定期更新学习计划以适应学生能力的变化。
  ......
  学习计划周期：{planning_period}天
  当前规划阶段：第{current_phase}阶段（共{total_phases}个阶段）

  请用中文回答，需要规划的学习需求是：{input}
metadata:
  type: 'PROMPT'
  version: academic_planner_agent.cn
```

反思与进化智能体（Reflection & Evolution Agent）：

作为系统的“质量监督员”和“进化引擎”，它会定期分析所有智能体的交互日志和用户反馈。通过反思（Reflection）机制，它能发现知识库的缺失、答案的不准确或规划的不合理之处，并自动触发知识库的更新流程或向管理员发出优化建议，实现系统的持续自我进化。

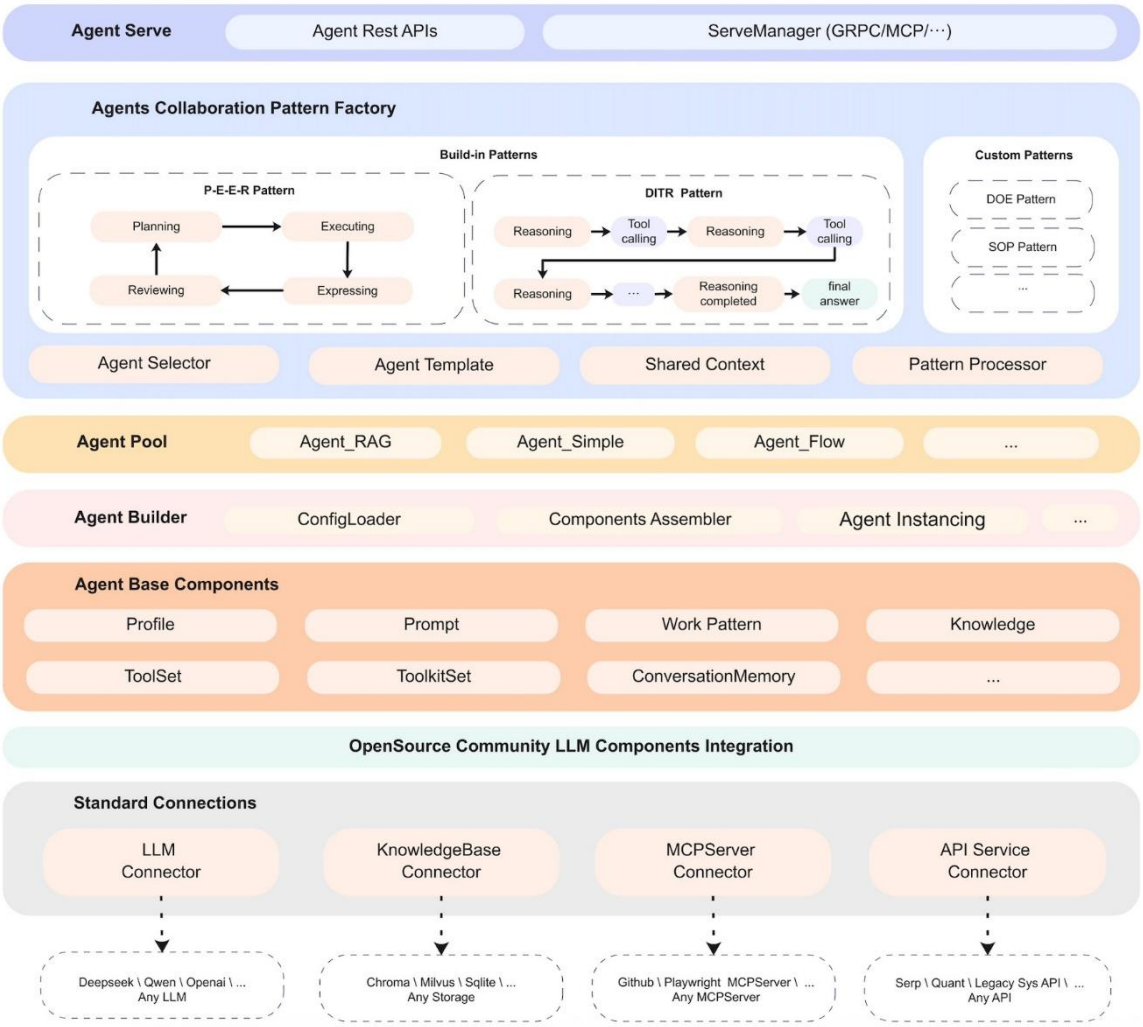


与蚂蚁集团生态的融合

多智能体-知识库协作系统作为“AI 智能 • 学习搭子”解决方案的智能中枢，其与蚂蚁集团生态的深度融合，核心体现在对 AgentUniverse 框架的深度应用与扩展上。该系统不仅继承了蚂蚁集团在复杂金融业务中锤炼出的多智能体协同能力，更将其创新性地迁移至教育场景，实现了从“静态知识问答”到“动态知识服务”的跃迁

AgentUniverse 已在蚂蚁内部的风控、客服、投研等场景中得到充分验证，具备处理复杂逻辑、保障服务稳定性的能力。将其应用于教育场景，确保了知识协作流程的严谨性与容错性，避免因“幻觉”或逻辑混乱误导学生。

AgentUniverse 框架提供的“模式工厂”机制，允许我们快速定义和组合不同的智能体协作模式。例如，针对“学生提问”这一场景，可动态组合“检索→生成→反思”三步协作模式；针对“学习规划”场景，则组合“评估→规划→推送”模式，实现高度场景化的智能服务。



AgentUniverse 天然支持遵循 MCP/A2A 的标准化智能体通信协议，使我们的知识库智能体、规划智能体等均可作为可插拔、可复用、可被发现的服务单元，未来可无缝接入蚂蚁集团内外的其他教育应用（如数字马力的助教平台、蚂蚁公益的学习资源平台），构建开放的教育智能体生态。

除 AgentUniverse 框架外，系统在整体架构上全面融入蚂蚁技术生态：

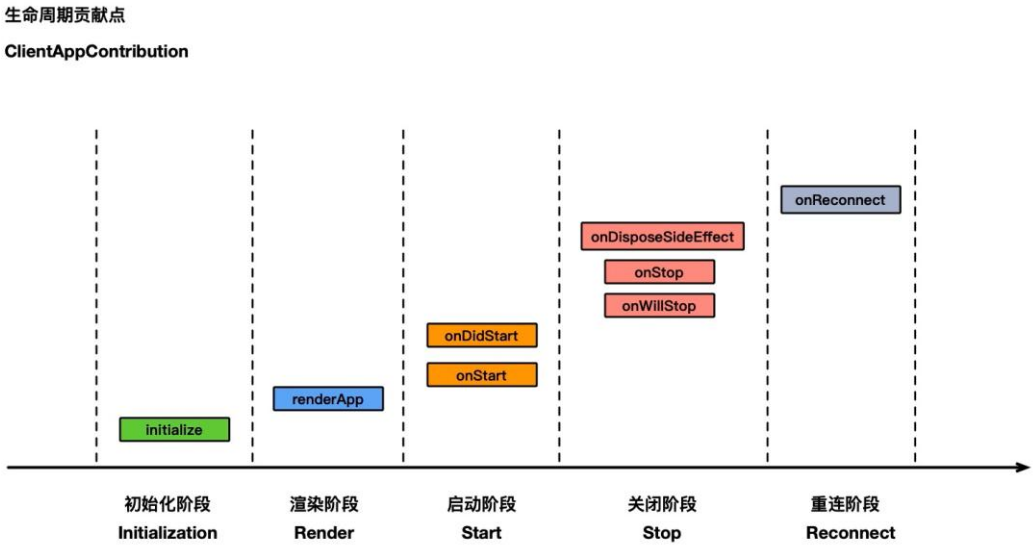
所有智能体均以 Qwen 系列大模型（如 Qwen3-4B）为推理引擎，该模型由阿里通义实验室研发，是蚂蚁 AI 生态的核心组件。统一的模型底座确保了智能体间语义理解的一致性，并便于利用蚂蚁在模型安全、价值观对齐等方面的成熟机制；智能体间的通信部署在微服务+ Service Mesh 架构之上，通过蚂蚁推荐的 OAuth2 + JWT 机制进行认证授权，并采用 HTTPS 协议进行加密传输，确保知识交互过程中的数据安全与隐私合规，符合教育行业对数据安全的严苛要求。

系统中的“学习规划智能体”可直接对接数字马力在产教融合中积累的行业技能图谱与岗位能力模型。当学生进入“二阶段”（就业导向）学习时，智能体能基于真实岗位需求（如“Java 后端开发工程师”），动态生成包含技术栈、项目经验、面试题库在内的个性化学习路径，有效弥合高校教学与市场需求的鸿沟；并可作为核心能力模块，打包接入蚂蚁公益平台，以技术手段推动教育公平。

多学科客制化评估系统

本系统是“AI 智能 • 学习搭子”解决方案中实现精准诊断与效果闭环的关键一环。它并非一个通用的题库系统，而是针对不同学科（如计算机、化学）的独特知识结构和能力评估标准，提供高度客制化的测评与反馈服务。系统旨在解决“学得怎么样”和“哪里不会”的核心问题，为数字人伴学系统和学习规划提供数据驱动的决策依据，契合赛题中“一阶段（校园考试）”与“二阶段（就业技能）”的双场景需求。

我们以高校中计算机学科的实际教学为切入点，构建了首个客制化评估模块，并为其他学科的快速扩展奠定了技术基础；其采用蚂蚁集团开源的 OpenSumi IDE 框架作为核心载体。OpenSumi 是一个高性能、高可定制的双端（Web/Electron）IDE 开发框架，能够快速构建出云端或桌面端的集成开发环境。



基于 OpenSumi 提供的前端 IDE 框架，团队集成了隔离的代码执行沙箱环境。学生可以直接在浏览器中编写、运行和调试代码，无需担心环境配置问题；同时，团队也开发了与项目深度集成的代码自动评测系统。该引擎不仅能判断代码的正确性（通过单元测试），还能从代码风格、算法效率、内存占用等多个维度进行综合评分，提供远超“对/错”的精细化反馈；同时，结合大语言模型（LLM），对学生的错误代码进行智能分析，不仅能指出错误原因，还能生成针对性的修复建议和相关知识点链接，实现“以评促学”，让编程学习不止停留在“纸上谈兵”。其核心代码示例如下：

```
import { AppRenderer, EditorRenderer } from '@codeblitzjs/ide-core';
function App() {
    return <AppRenderer appConfig={{}} />
}
function Editor() {
    return <EditorRenderer appConfig={{}} />
}

```

针对高校计算机学科在线评测系统（Online Judge, OJ）的实际需要，声像科技团队也搭建了对应的代码评测沙箱，其核心代码示例如下：

```
@Override
public ExecuteCodeResponse executeCode(ExecuteCodeRequest executeCodeRequest) {
    List<String> inputList = executeCodeRequest.getInputList();
    String code = executeCodeRequest.getCode();
    String language = executeCodeRequest.getLanguage();
//        1. 把用户的代码保存为文件
    File userCodeFile = saveCodeToFile(code);
//        2. 编译代码，得到 class 文件
    ExecuteMessage compileFileExecuteMessage = compileFile(userCodeFile);
    System.out.println(compileFileExecuteMessage);
    // 3. 执行代码，得到输出结果
    List<ExecuteMessage> executeMessageList = runFile(userCodeFile, inputList);
//        4. 收集整理输出结果
    ExecuteCodeResponse outputResponse = getOutputResponse(executeMessageList);
//        5. 文件清理
    boolean b = deleteFile(userCodeFile);
    if (!b) {
        log.error("deleteFile error, userCodeFilePath = {}",
userCodeFile.getAbsolutePath());
    }
    return outputResponse;
}
```

对用户提交的代码进行编译与执行，其核心代码示例如下：

```
public List<ExecuteMessage> runFile(File userCodeFile, List<String> inputList) {
    String userCodeParentPath = userCodeFile.getParentFile().getAbsolutePath();
    List<ExecuteMessage> executeMessageList = new ArrayList<>();
    for (String inputArgs : inputList) {
        String runCmd = String.format("java -Xmx256m -Dfile.encoding=UTF-8 -cp %s Main
%s", userCodeParentPath, inputArgs);
        try {
            Process runProcess = Runtime.getRuntime().exec(runCmd);
            new Thread(() -> {
                try {
                    Thread.sleep(TIME_OUT);
                    runProcess.destroy();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            }).start();
            ExecuteMessage executeMessage =
ProcessUtils.runProcessAndGetMessage(runProcess, "运行");
            System.out.println(executeMessage);
            executeMessageList.add(executeMessage);
        } catch (Exception e) {
            throw new RuntimeException("执行错误", e);
        }
    }
    return executeMessageList;
}
```

跨学科扩展框架：

针对其他学科的实际需要，声像科技团队也搭建了相关的学科题库，以更好服务于团队所研发的多智能体-知识库协同系统与实际教育教学过程中的客观需要。

面向部分学科，我们也搭建了相应的跨学科拓展框架。

数学：团队通过集成视觉大模型（Qwen-VL）实现数学公式识别和符号计算；学生可以手写或输入数学公式，系统能自动解析并评估其推导过程的逻辑与正确性。

化学：团队基于 Three.js 的前端 3D 界面构建分子结构的可视化模型；使得教师能够在实际授课中全面展现分子结构，实现可视化的教学过程。

通用技术：团队结合教学过程中的授课 PPT、教科书 PDF 等多元信息，对 Qwen3 大模型进行微调（Fine-Turning），实现题目对应相关知识的溯源过程。

与蚂蚁集团生态的融合

声像科技团队依据高校计算机学科的实际教学需要，结合蚂蚁集团的 OpenSumi 前端，开发了客制化的编程 IDE；默认集成 Ant Design 组件库，确保评估界面与蚂蚁系产品风格统一，操作逻辑符合用户习惯，降低高校师生的学习成本。

系统严格遵循企业文档提出的安全规范；通过 OAuth2 + JWT 实现用户认证，所有代码执行均在隔离的安全沙箱中进行，防止恶意代码注入；数据传输全程采用 HTTPS 加密，保障学生代码与评估结果的数据信息安全。

评估系统的评测引擎、题库管理、结果分析等模块可拆分为独立微服务，通过 Service Mesh 进行治理，与整体“AI 智能·学习搭子”解决方案的技术架构保持一致，实现 SaaS（Software as a Service）便于未来能力复用与弹性扩展。

评估系统可对接数字马力产教融合体系；可接入高校共建的课程标准与技能图谱，实现客制化服务定制。例如，针对“Java 后端开发”岗位，系统可自动生成包含 Spring Boot、MySQL、Redis 等技术栈的综合项目题，并依据企业用人标准进行多维度评分，实现“学即所用、评即所聘”。

数据驱动学习闭环；用户在使用评估系统时的结果（如“算法效率低”、“异常处理缺失”）将实时同步至多智能体-知识库系统，触发学习规划智能体生成针对性复习计划；同时，该数据也会反馈给数字人伴学系统，由数字人主动关怀并引导学生查漏补缺，形成“评估—规划—陪伴—再评估”的完整闭环。

通过以上所述的多学科客制化评估系统，声像科技团队不仅实现了技术上的工程化、标准化与可扩展性，更在业务层面深度融入了数字马力的产教融合实践与蚂蚁集团的开发者工具生态。这种融合将企业级的工程能力创造性地应用于教育评估领域，为“AI 智能·学习搭子”项目提供了真实、精准、可衡量的能力验证手段，有力支撑了“从校园到职场”的双阶段学习目标，更彰显着服务外包大赛“来源实际需求，鼓励创新应用”的核心原则。

游戏化学习平台及评估系统

本系统是“AI 智能 • 学习搭子”解决方案中面向入门级学习者（如小学生）和兴趣驱动型学习场景（如语言学习）的核心模块。其核心价值在于，通过将游戏化（Gamification）设计理念深度融入学习过程，有效解决低龄或初学者“学习动力不足、注意力难以集中、缺乏即时正向反馈”的痛点，让学习变得像游戏一样有趣、有挑战、有成就感，从而激发内在学习动机，为后续的进阶学习打下坚实基础。

技术架构与游戏化设计

我们以小学英语为切入点，构建了首个游戏化学习模块，并为其他语言学科（如日语）的快速扩展提供了范式。

技术基座：采用蚂蚁集团主导开发的 Ant Design 企业级 UI 设计系统作为前端开发框架。Ant Design 不仅提供了丰富、美观、一致的 UI 组件，其设计价值观中也蕴含着“创造快乐工作”的理念，这与我们的游戏化目标高度契合。我们利用其响应式布局能力，确保平台在平板和电脑上均能提供流畅、沉浸的用户体验。

核心游戏化机制：

角色与成长体系：学生创建自己的虚拟角色（Avatar），通过完成学习任务（如单词拼写、听力练习、口语对话）获得经验值（XP）和金币，用于解锁新装扮、新技能及进入更高级别的学习关卡。

即时反馈与奖励：借鉴“百词斩”等成功产品的经验，系统在学生完成每个微小任务后，立即给予视觉（动画、徽章）和听觉（音效）的正向反馈，强化学习行为。

挑战与任务系统：将学习目标分解为一系列由易到难的“每日挑战”和“主题任务”，如“一周内掌握 20 个动物单词”。任务设计紧密结合教学大纲，确保游戏性与教育性的统一。

智能评估与反馈：

嵌入式评估：评估不再是独立的考试，而是完全融入游戏流程中。例如，在一个“餐厅点餐”的角色扮演游戏中，系统会通过语音识别（ASR）和自然语言理解（NLU）技术，实时评估学生的发音准确度和句型使用是否恰当。

数据驱动的个性化：平台会记录学生在游戏中的所有行为数据（如错误类型、反应时间、重试次数），并利用这些数据动态调整后续任务的难度，确保挑战始终与学生的能力相匹配，维持其“心流”状态。

与蚂蚁集团生态的融合

赋能普惠教育：本系统适合通过蚂蚁公益平台进行推广。我们可以将其打包为一个公益项目，例如“乡村儿童英语启蒙计划”，通过支付宝公益频道的“行为公益”（如用户捐步数）或直接捐赠，为资源匮乏地区的学校和学生提供免费的、高质量的游戏化英语学习资源，践行数字马力的普惠教育使命。

融入教育机构合作：与数字马力已合作的高校及中小学合作，将本平台作为其英语教学的官方辅助工具，教师可以在后台查看班级整体和学生个体的学习数据报告，实现精准教学。