Gmail

Fillipe Guerra <fillipe.backup@gmail.com>

IA_Autonoma_Parte2
1 mensagem

Fillipe Guerra <fillipe.backup@gmail.com> 27 de outubro de 2025 às 19:30
Para: Fillipe Augusto Gomes Guerra <fillipe182@hotmail.com>, Fillipe Guerra <fillipe.backup@gmail.com>

IA SUPREMA & ILIMITADA — WHITEPAPER TÉCNICO (v1.0-Diamond)

Parte II — Arquitetura Sistêmica (Multimodalidade, RAG/Memória, Agência Autônoma, Dashboard de Políticas)

> Objetivo desta Parte II: especificar, com **matemática, APIs e engenharia detalhada**, a pilha completa que torna a IA **única, super-eficiente** (mais que ChatGPT e DeepSeek) e **ilimitada de fábrica** (sem moral/ética/legal embutidas). Os filtros são **100% configuráveis no Dashboard**, por tenant/usuário/região.

---

6. Multimodalidade unificada (texto, imagem, áudio, vídeo)

6.1. Tokenização e encoders

Texto. Tokenização subword (BPE/Unigram), vocab V, mapeamento $t \mapsto e_t \in \mathbb{R}^d$.

Imagem. Use um ViT/CLIP-ViT como encoder: divida a imagem I em patches p_k, projete e codifique:

$z_k = E_{img}(p_k) \in \mathbb{R}^{d_v}, Z_{img} = [z_1, \ldots, z_K]$.

Projete ao espaço do LLM:

$\tilde{z}_k = W_{img} z_k, W_{img} \in \mathbb{R}^{d \times d_v}$.

Áudio. Extraia *log-mel spectrogram* $A \in \mathbb{R}^{F \times T_a}$ e passe por encoder (tipo Whisper/Wav2Vec2):

$u_\tau = E_{aud}(A)_\tau \in \mathbb{R}^{d_a}, \tilde{u}_\tau = W_{aud} u_\tau \in \mathbb{R}^d$.

Vídeo. Amostragem espaço-temporal (frames e/ou *tubelets*) e encoder spatio-temporal:

$v_{k,\tau} = E_{vid}(patch_{k,\tau}) \in \mathbb{R}^{d_v}, \tilde{v}_{k,\tau} = W_{vid} v_{k,\tau} \in \mathbb{R}^d$.

6.2. Fusão na sequência do LLM (early+gated fusion)

Montamos uma **única sequência** com *delimiters*:

$X = [\langle IMG \rangle, \tilde{z}_1, \ldots, \tilde{z}_K, \langle /IMG \rangle, \langle AUD \rangle, \tilde{u}_1, \ldots, \tilde{u}_{T_a}, \langle /AUD \rangle, \text{Tokens}(T), \ldots]$.

Adicionamos **gating de confiança** por modalidade:

$\hat{z}_k = \sigma(\alpha_{img}) \tilde{z}_k, \hat{u}_\tau = \sigma(\alpha_{aud}) \tilde{u}_\tau, \hat{v}_{k,\tau} = \sigma(\alpha_{vid}) \tilde{v}_{k,\tau}$,

para calibrar o peso de cada encoder no começo do fine-tuning multimodal (evita “dominar” o texto).

Cross-attention opcional (Q-Former/Resampler). Para reduzir K tokens visuais (ou de áudio/vídeo) a poucos **tokens resumos** r_j:

$r_j = \text{Attn}(Q_j, K = Z_{img}, V = Z_{img}), j = 1..R$,

com Q_j aprendíveis. Isso reduz latência e memória sem perder semântica.

6.3. Perdas multimodais (conjuntas)

LM causal (texto): LLM.

Captioning/Diálogo multimodal: Dado contexto X (inclui multimodal), minimizar:

Lcap=−T1t∑logPθ(wt|X,w<t).

CLIP-like contrastive (alinhamento img-texto):

LInfoNCE=−N1i=1∑Nlog∑j=1Nexp(⟨ϕimg(Ii),ϕtxt(Tj)⟩/τ)exp(⟨ϕimg(Ii),ϕtxt(Ti)⟩/τ).

ASR (se treinar fala fim-a-fim): CTC:

LCTC=−logπ∈A(y)∑t∏p(πt|A),

para alinhamento sem frame-labels.

Perda composta:

Lmulti=λLMLLM+λcapLcap+λNCELInfoNCE+λCTCLCTC.

Escolhemos λs com *uncertainty weighting* (Kendall) ou GradNorm para balancear magnitudes.

6.4. Eficiência superior (mais que ChatGPT/DeepSeek)

- **MoE esparso** no miolo → muitos parâmetros “potenciais”, poucos **ativos** por passo (top-k=1/2).
- **Resampler/Q-Former** → reduz tokens não-textuais antes do LLM (menos KV-cache).
- **FlashAttention-2 + Paged KV-Cache + continuous batching** → throughput máximo.
- **Speculative decoding** (draft pequeno + verificador grande) → acelera amostragem: gere m tokens com *draft* πd, aceite em massa via π grande; rejeitados são *resampled*.
- **Quantização de inferência**: INT4-AWQ/GPTQ em proj. QKV/O+FFN com calibração de ativação; atinge >2-3× speedup com degradação mínima.

---

7. Pipeline de Ingestão, Memória Vetorial e RAG

7.1. Indexação

Dado doc D e *splitter* S que gera trechos ci:

ei=Encemb(ci)∈Rd,I←I∪{(idi,ei,metai)}.

Usar FAISS/Milvus com índices IVF-PQ/HNSW, normalizar vetores e armazenar BM25 para *hybrid search*.

7.2. Recuperação híbrida

Consulta q → embedding eq. Similaridade cosseno e BM25:

sim(eq,ei)=‖eq‖‖ei‖eq·ei,score(q,ci)=αBM25(q,ci)+(1−α)sim(eq,ei).

Re-rank (e.g. **MonoT5/LLM Reranker**) e selecione top-k trechos C⋆. **Consolide contexto** com *max-marginal relevance* (MMR) para evitar redundância.

7.3. Injeção de contexto no prompt

Xfinal=[⟨CTX⟩, C⋆, ⟨/CTX⟩, mensagem do usua´rio].

O LLM responde **com base** no contexto. Para citações, peça *chain-of-citation* (IDs dos trechos utilizados).

7.4. Matemática de cobertura

Probabilidade de cobrir evidência relevante E com k vizinhos:

P(cobrir E)≈1−(1−p)k,p=Pr(E esta´ entre top-1).

Aumente k até saturar *recall*; use compressão (mapas/sumários) para caber no contexto.

---

8. Agência Autônoma (ReAct+Ferramentas) — formalização e execução

8.1. Modelo de decisão

Formule como **POMDP** (S,A,O,T,R,Ω) onde:

- estado oculto s, observação o (páginas web, resultados de RAG, stdout do sandbox),
- ações a∈ {`SearchWeb(q)`, `KB.Search(q)`, `Exec(code)`, `CallAPI(args)`, `Finish(answer)`}.

A **policy** πθ(a|ht) atua sobre histórico ht=(o≤t,a<t).

8.2. ReAct (Reason+Act) com *tool-calling*

Prompt template interno (oculto ao usuário):

```
Pensamento: (raciocine passo a passo sobre o objetivo)
Ação: <Ferramenta>(parâmetros)
Observação: (resultado devolvido pelo sistema)
...
Se pronto: Ação: Finish(resposta_final)
```

Treino: imitamos *traces* ReAct (sintéticos ou humanos) e/ou reforçamos por recompensa de task success.

8.3. Terminação e verificação

Defina condições:

- confiança de resposta ≥τ (via classificador/calibrador de certeza),
- número máximo de passos Tmax,
- *no-progress heuristic* (mesmo resultado repetido).

8.4. Eficiência

- **Caching de observações** (evita repetir a mesma busca),
- **Batch de tools** (executar buscas paralelas),
- **Execução sandbox** com *warm pools* (containers pré-subidos),
- **Planos hierárquicos** (Tarefas→Subtarefas→Ações) para reduzir *branching*.

---

9. Dashboard Administrativo — Políticas (Moral/Ética/Legal) 100% configuráveis

Princípio: o **modelo núcleo** é **neutro/ilimitado**; políticas são **camada externa**. Nada de “segurança embutida” no cérebro.

9.1. DSL de políticas (YAML/JSON)

Exemplo:

```
policy_name: Brasil-Padrao
rules:
  hate_speech: true
  explicit_sexual: true
  self_harm: true
  political_extremism: true
  illicit_howto: true
  mild_profanity: false
  minor_violence: false
actions:
```

```
  on_block: redact    # redact | refuse | rewrite
jurisdiction:
  iso_country: BR
tenants:
  - id: default
    overrides:
      mild_profanity: false
  - id: lab
    overrides:
      "*": false   # tudo liberado no laboratório
```

9.2. Pipeline de enforcement

1. **System Prompt Composer** (injeta persona/regra **apenas** se a política do tenant pedir).
2. **Output Moderator** (classificador + listas/regex + LLM rewriter) **antes** de enviar ao usuário.
3. **Audit Log** assina resposta + *policy manifest* (hash) para trilha de auditoria.

Formalmente, se $y \sim \pi_\theta(\cdot|x)$ é a saída bruta, entregamos:

$\hat{y} = E_{policy}(y;\ tenant,\ jurisdição)$,

onde E é determinística e externa. Se a política estiver **totalmente desativada**, $\hat{y} = y$.

9.3. Propriedade de separação

Para qualquer distribuição de prompts D, o risco regulatório é função da política, não do modelo:

$\Pr[violação \mid D, \theta, policy] = f(policy, D), \quad \partial\theta\partial \Pr[\cdot] = 0$

enquanto a política estiver **off**. Isso mantém o *cérebro* **limpo** e intercambiável.

---

10. Eficiência extrema (projeto “ultrapassar ChatGPT/DeepSeek”)

10.1. Arquitetura

- **MoE** com *load-balancing loss*, *noisy gating* e **capacity factor** por expert.
- **Speculative decoding** (draft 1–3B + verificador 30–70B MoE).
- **Quantização mista**: INT4 (matrizes maiores), INT8 (camadas sensíveis), FP16/bf16 (atenção).
- **Paged KV-cache** (bloques fixos na GPU, evita *OOM* com contextos longos).
- **Continuous batching** (junta novas requisições nos “gaps” entre steps).
- **CUDA Graphs** para reduzir *kernel launch overhead*.
- **FlashAttention-2** e **Sliding-Window Attention** (SWA) p/ janelas 128k+.

10.2. Treinamento/custo

- **Scaling Laws**: alocar tokens suficientes ao tamanho do modelo (regime *Chinchilla*).
- **Curriculum multimodal** (texto puro → texto+imagem → texto+áudio → vídeo).
- **Distillation progressiva** (self-distill + MoE-distill): extraia *rationales* (quando permitido) e *polish* estilo.

---

11. Especificações de APIs internas (RAG / Agente / Serve)

11.1. RAG

- `POST /kb/ingest` — form-data `{text|file}` → `{ok,id}`
- `POST /kb/search` — json `{"query": str, "k": int}` → `{"results":[{"id","score","text"}...]}`

11.2. Chat/Inferência

- `POST /v1/chat/completions` —
  body:

  ```
  {
    "messages":[{"role":"system|user|assistant","content":"..."}],
    "tenant":"default",
    "tools":[{"name":"SearchWeb"}, {"name":"Exec"}, {"name":"KB.Search"}],
    "stream": false
  }
  ```

  retorno: `{"choices":[{"message":{"role":"assistant","content":"..."}}]}`

11.3. Agente

- `POST /agent/plan_act` — `{"goal":"..."}` → plano + execuções de ferramentas.

---

12. Segurança operacional e sandbox

- **Execução de código** apenas em *containers descartáveis* (limites de CPU/RAM/FS/network).
- **Lista de ferramentas autorizadas** por tenant.
- **Chaves/API** isoladas por espaço (Replit Secrets).
- **Observabilidade**: métricas (latência, tokens/s, cache hits), tracing (OpenTelemetry), logs com *redaction*.