Relatório final de atividades

Projeto: SAVITS: Sistema de Avaliação e Validação de Impactos em Tecnologias Sociais

Coordenador do Projeto:	**Washington Luís Ribeiro de Carvalho Segundo**		
Nome do(a) pesquisador(a):	**Fábio Diniz Rossi**		
Vigência da Bolsa de Pesquisa:	**08/06/2026**	**Até:**	**08/09/2026**
Encerramento em:	**08/09/2026**		
Ref. Fundação:	**32725**	**Ref. Ibict:**	**008/2026-81**

Descrição das Atividades da Bolsa de Pesquisa:

Estudar, analisar e propor modelos de arquitetura digital para suporte à infraestrutura do SAVITS, considerando servidores, containers (Docker/Kubernetes), redes e ambientes experimentais de desenvolvimento, homologação e produção, com foco em escalabilidade, segurança, rastreabilidade e aderência aos objetivos técnicos do projeto. Meta 2; Estudar métricas, métodos e procedimentos de avaliação de disponibilidade, desempenho, segurança e observabilidade da infraestrutura digital da plataforma, propondo diretrizes metodológicas para uso de alertas, logs centralizados, rotinas de backup, resiliência operacional e interoperabilidade por meio de APIs abertas. Meta 2: Elaborar relatório técnico final da bolsa, consolidando resultados, atividades desenvolvidas e contribuições metodológicas da pesquisa.

1. Introdução

O projeto SAVITS - Sistema de Avaliação e Validação de Impactos em Tecnologias Sociais - propõe a construção de uma plataforma de inteligência analítica e preditiva voltada à avaliação de impactos associados a tecnologias sociais e políticas públicas. A proposta articula *Business Intelligence*, Aprendizado de Máquina e modelos de linguagem generativos em um mesmo ecossistema, de modo que diferentes formas de informação possam ser coletadas, processadas, analisadas e apresentadas aos pesquisadores e gestores. No documento orientador do projeto, os modelos de linguagem são associados especialmente às funções de interpretação em linguagem natural, análise semântica, sumarização e extração de temas em relatórios qualitativos e produções textuais. Isso significa que a escolha tecnológica não deve ser orientada apenas pelo desempenho genérico de um modelo, mas pela capacidade de atender a essas tarefas dentro das condições de governança, privacidade, transparência, reprodutibilidade e soberania informacional previstas para o SAVITS.

A definição de uma estratégia de Inteligência Artificial generativa exige distinguir dois problemas que frequentemente são tratados como se fossem equivalentes. O primeiro é a escolha do modelo de linguagem propriamente dito: qual família utilizar, qual quantidade de parâmetros é suficiente, qual precisão numérica adotar e qual mecanismo de execução será empregado. O segundo é a forma de adaptar esse modelo ao domínio e às necessidades do sistema. Um modelo pode ser utilizado sem qualquer modificação, pode receber instruções e exemplos no contexto, pode ser conectado a bases externas por meio de *Retrieval-Augmented Generation*, pode ser ajustado com LoRA ou QLoRA, pode passar por *fine-tuning* completo, pode ter seu pré-treinamento continuado em um *corpus* específico ou, em uma situação mais extrema, pode ser treinado do zero. Cada alternativa impõe requisitos muito diferentes de hardware, tempo, dados, custo operacional e capacidade técnica da equipe.

Para o SAVITS, essa decisão possui consequência direta sobre a infraestrutura a ser adquirida. O projeto prevê explicitamente a identificação, aquisição e configuração de hardware e software para sustentação da plataforma. Uma infraestrutura dimensionada

apenas para executar um modelo quantizado de pequeno ou médio porte pode ser suficiente para inferência e RAG, mas insuficiente para ajuste de parâmetros. Por outro lado, adquirir desde o início um cluster destinado ao treinamento integral de grandes modelos pode representar custo elevado sem que exista, nesta etapa, evidência de necessidade. O desafio, portanto, é propor uma infraestrutura capaz de atender às demandas atuais, sustentar a experimentação científica e preservar uma trajetória razoável de expansão.

Este relatório apresenta uma análise técnica das principais alternativas para utilização e adaptação de modelos generativos no contexto do SAVITS, descrevendo suas características, infraestrutura mínima e recomendada, tempos aproximados de preparação ou treinamento e aderência às necessidades do projeto. A análise procura manter uma perspectiva pragmática: o objetivo não é construir um grande modelo por si só, mas estabelecer uma arquitetura de IA generativa que seja tecnicamente justificável, sustentável, auditável e compatível com a natureza institucional dos dados tratados pelo sistema.

2. Requisitos do SAVITS para o uso de modelos generativos

A arquitetura de modelos generativos deve ser derivada das necessidades do SAVITS e não do simples interesse em utilizar uma tecnologia específica. O projeto prevê a coleta e o tratamento de dados estruturados, semiestruturados e qualitativos, que alimentarão tanto os modelos preditivos tradicionais quanto os módulos de análise semântica. Os dados textuais poderão assumir formas diferentes, como relatórios, respostas abertas, registros produzidos por participantes, documentos metodológicos, resultados de avaliação e outros conteúdos associados ao programa piloto. O modelo de linguagem deverá, portanto, trabalhar como componente de uma infraestrutura de dados mais ampla, interoperando com bancos relacionais, bases documentais, mecanismos de recuperação de informação, módulos analíticos e interfaces de visualização.

A primeira função relevante é a análise semântica. Nesse contexto, o objetivo não é apenas localizar palavras ou expressões, mas identificar significados, relações, temas e regularidades em textos que podem ser heterogêneos e pouco padronizados. Um relato

pode tratar de acessibilidade, participação, barreiras, percepção de impacto ou dificuldades institucionais sem empregar exatamente os termos previamente definidos por um pesquisador. Modelos de linguagem são capazes de reconhecer esse tipo de proximidade semântica e produzir representações ou classificações úteis. Contudo, essa capacidade deve ser acompanhada de mecanismos de validação, pois a interpretação gerada pelo modelo é probabilística e não substitui automaticamente o julgamento metodológico.

A sumarização constitui outra aplicação central. Em vez de exigir que pesquisadores ou gestores leiam individualmente grandes volumes de texto, o sistema pode produzir sínteses preliminares, organizar pontos recorrentes e destacar elementos relevantes. Para o SAVITS, entretanto, uma síntese não deve ser tratada como evidência autônoma. O sistema deve manter a possibilidade de retornar aos trechos originais, de modo que a informação resumida possa ser conferida e auditada. O mesmo raciocínio vale para a extração de temas, entidades ou categorias: o resultado do modelo deve ser interpretado como uma camada analítica derivada dos dados, e não como substituto do dado original.

Outro requisito importante é a possibilidade de consulta em linguagem natural. Essa funcionalidade pode tornar a plataforma mais acessível para usuários que não dominem estruturas de banco de dados ou linguagens de consulta. Um gestor poderia perguntar, por exemplo, quais temas aparecem com maior frequência em determinado conjunto de relatos, quais documentos sustentam uma determinada interpretação ou que padrões foram identificados em um recorte temporal. Para que esse uso seja confiável, o modelo não deve responder apenas a partir de conhecimento genérico adquirido durante seu pré-treinamento. Ele precisa ser capaz de acessar, de forma controlada, as fontes institucionais relevantes e de indicar quais evidências foram utilizadas na resposta.

A preocupação com soberania informacional torna especialmente importante a possibilidade de execução local ou institucional. O projeto SAVITS estabelece protocolos de custódia, anonimização, segurança e governança dos dados e enfatiza a necessidade de reduzir dependências tecnológicas externas. A execução de modelos em infraestrutura própria reduz a exposição de conteúdos a serviços de terceiros e permite maior controle

sobre logs, políticas de acesso, versões de modelo e ciclo de vida dos dados. Isso não elimina os riscos de segurança, mas transfere para a instituição a capacidade de implementar controles mais coerentes com sua política de governança.

A transparência e a reprodutibilidade também devem orientar a escolha. Um sistema que utiliza uma API externa pode sofrer alterações de comportamento sem que a equipe tenha controle sobre a versão efetiva do modelo. Em uma infraestrutura local, é possível registrar com precisão a versão dos pesos, o *tokenizer*, a quantização, os parâmetros de geração, o prompt de sistema, a base de recuperação, o modelo de *embeddings* e os critérios utilizados para seleção de contexto. Essa capacidade de versionamento é particularmente compatível com a exigência do SAVITS de manter rastreabilidade, auditabilidade e reprodutibilidade científica.

3. Estratégias para implantação e adaptação de modelos de linguagem

3.1 Modelos pré-treinados, prompting e aprendizagem em contexto

A estratégia mais simples consiste em selecionar um modelo previamente treinado e executá-lo sem alterar seus pesos. Famílias contemporâneas de modelos abertos ou com pesos disponíveis, como Qwen, Llama, Gemma, Mistral e Phi, oferecem versões com diferentes quantidades de parâmetros e diferentes características de licença e uso. A execução local pode ser realizada por ferramentas como Ollama, llama.cpp, Transformers ou vLLM. A principal vantagem dessa abordagem é a rapidez: após o download e a configuração do ambiente, o modelo pode ser testado imediatamente. Não existe treinamento, e o custo computacional está concentrado na inferência.

A viabilidade local depende fortemente da representação numérica. Um modelo armazenado em FP16 ou BF16 utiliza aproximadamente dois bytes por parâmetro apenas para os pesos. Isso significa que um modelo de 7 bilhões de parâmetros pode demandar aproximadamente 14 GB somente para a carga dos pesos, sem considerar o contexto, buffers e estruturas auxiliares. A quantização reduz esse requisito ao representar os pesos em 8 ou 4 bits. Em muitas situações, um modelo 7B ou 8B quantizado em 4 bits pode ser

executado em uma GPU de 8 a 16 GB de VRAM, embora uma configuração com maior memória ofereça margem para contexto mais amplo, maior concorrência e uso de modelos maiores.

O comportamento do modelo pode ser modificado sem treinamento por meio de *prompt engineering*. Um system prompt pode estabelecer que as respostas devem ser técnicas, que informações sem evidência devem ser explicitamente sinalizadas, que determinadas categorias devem ser utilizadas ou que a saída deve seguir uma estrutura definida. Essa forma de adaptação é simples, reversível e praticamente sem custo computacional adicional além do próprio tamanho do contexto. Para o SAVITS, ela deve constituir a primeira linha de experimentação, pois permite avaliar até que ponto as capacidades necessárias já existem no modelo base.

Uma extensão é o *few-shot prompting*, dentro do conceito mais amplo de *In-Context Learning*. Nesse caso, o modelo recebe exemplos de entrada e saída antes da tarefa real. É possível, por exemplo, fornecer dois ou três relatos acompanhados de classificações temáticas aprovadas por especialistas e, em seguida, solicitar que um novo relato seja analisado no mesmo padrão. Essa técnica pode produzir um comportamento que superficialmente lembra um modelo ajustado, mas nenhum peso é alterado. Os exemplos permanecem no contexto e precisam ser reenviados a cada nova requisição. Sua principal limitação é, portanto, o consumo da janela de contexto e o aumento da latência à medida que o prompt cresce.

Do ponto de vista metodológico, modelo pré-treinado, *system prompt e few-shot* devem ser tratados como baselines. Antes de realizar qualquer treinamento, é necessário medir o que pode ser obtido apenas com configuração de prompt. Caso o desempenho seja suficiente para determinadas tarefas, não existe justificativa técnica para introduzir maior complexidade.

3.2 Retrieval-Augmented Generation - RAG

Retrieval-Augmented Generation (RAG) é particularmente relevante para o SAVITS porque separa a capacidade linguística do modelo do conhecimento institucional utilizado na resposta. Em uma arquitetura RAG, os documentos e dados não precisam ser

incorporados aos pesos do modelo. Eles permanecem em uma base externa e são recuperados no momento da consulta. A pergunta do usuário é utilizada para localizar trechos relevantes; esses trechos são então fornecidos ao modelo como contexto para a geração da resposta.

A construção de um RAG normalmente envolve preparação documental, divisão em trechos menores, geração de *embeddings* e indexação. Os *embeddings* representam textos em um espaço vetorial no qual conteúdos semanticamente semelhantes tendem a ocupar posições próximas. A consulta do usuário também é convertida em vetor, permitindo selecionar os trechos mais relacionados. Em uma implementação mais sofisticada, a busca vetorial pode ser combinada com busca lexical, filtros por metadados e reranking por modelos especializados.

Essa arquitetura oferece duas vantagens especialmente importantes. A primeira é a atualização do conhecimento sem retreinamento. Se um novo relatório ou documento for incorporado à plataforma, ele pode ser processado, indexado e disponibilizado ao modelo sem que os pesos sejam modificados. A segunda é a rastreabilidade. Como o sistema conhece quais trechos foram recuperados, a interface pode apresentar as fontes utilizadas e permitir que o usuário confronte a resposta com os dados originais.

O próprio contexto institucional do projeto reforça essa opção. O documento do SAVITS cita o Pinakes.ia como um protótipo do Ibict que utiliza Recuperação Aumentada por Geração para combinar modelos de linguagem e mecanismos de busca em bases externas, ampliando a capacidade de resposta sem necessidade de retreinamento. Essa experiência prévia reduz o risco de adoção e favorece a reutilização de conhecimentos já existentes na instituição.

O RAG também pode ser combinado com grafos de conhecimento. A recuperação vetorial é eficiente para identificar proximidade semântica, mas nem sempre representa explicitamente relações entre entidades. Grafos podem registrar conexões formais entre pessoas, instituições, projetos, indicadores, documentos e outros elementos do ecossistema. Em uma arquitetura híbrida, a pergunta pode consultar simultaneamente um índice vetorial e um grafo de conhecimento, fornecendo ao modelo tanto trechos textuais quanto relações estruturadas. Essa possibilidade é coerente com as orientações

do SAVITS para interoperabilidade, *Linked Open Data* e organização semântica da informação.

3.3 Fine-tuning eficiente e especialização do comportamento

Nem toda limitação de um modelo pode ser resolvida apenas com RAG. Ele fornece informação, mas não necessariamente ensina ao modelo como executar uma tarefa especializada. Se o SAVITS precisar que o modelo produza análises segundo uma taxonomia específica, respeite um protocolo metodológico recorrente ou gere saídas estruturadas com elevado grau de consistência, pode ser necessário ajustar seu comportamento.

O *Supervised Fine-Tuning* utiliza exemplos de entrada e saída desejada para atualizar o modelo. Um conjunto de treinamento poderia conter textos analisados por especialistas e suas respectivas classificações, resumos ou estruturas de saída. No fine-tuning completo, todos ou grande parte dos pesos são atualizados. Isso exige memória significativamente maior que a inferência, porque o treinamento precisa manter pesos, gradientes, ativações e estados do otimizador.

LoRA, ou *Low-Rank Adaptation*, reduz esse custo ao introduzir pequenas matrizes treináveis em determinadas camadas do modelo e manter o restante dos parâmetros congelado. O resultado é um adaptador muito menor que o modelo original. QLoRA amplia esse princípio carregando o modelo base de forma quantizada, normalmente em 4 bits, enquanto os adaptadores são treinados em precisão adequada. Essa combinação tornou possível realizar ajustes de modelos de bilhões de parâmetros em GPUs de uma única *workstation*.

Para o SAVITS, LoRA e QLoRA são especialmente atraentes porque oferecem especialização real sem exigir um cluster de treinamento. Um modelo 7B ou 8B pode ser ajustado com uma GPU de aproximadamente 24 GB de VRAM em muitas configurações, enquanto modelos de 13B ou 14B podem demandar 24 a 48 GB, dependendo do comprimento das sequências, batch size, quantização e técnica utilizada.

RAG e QLoRA não devem ser vistos como alternativas excludentes. A combinação pode ser mais adequada: o QLoRA ajusta o comportamento e o RAG fornece o

conhecimento atualizado. Isso evita tentar incorporar fatos dinâmicos aos pesos e, ao mesmo tempo, permite adaptar o modelo ao formato e às tarefas específicas do SAVITS.

3.4 Estratégias de maior complexidade

Existem ainda alternativas mais exigentes. *Preference tuning*, com técnicas como DPO, utiliza pares de respostas preferidas e rejeitadas para alinhar o comportamento do modelo. *Continued pretraining* ou *Domain-Adaptive Pretraining* continua o pré-treinamento sobre um corpus especializado e pode produzir maior familiaridade com o vocabulário e os padrões de um domínio, mas demanda muito mais dados e capacidade computacional do que QLoRA. *Knowledge distillation* utiliza um modelo maior como teacher para auxiliar no treinamento de um modelo menor, estratégia que pode ser útil caso o projeto deseje, no futuro, disponibilizar um SLM altamente especializado.

O treinamento de um modelo do zero representa a alternativa mais onerosa. Além da infraestrutura, exige construção de corpus, tokenização, pré-treinamento, *instruction tuning* e etapas posteriores de alinhamento. É possível treinar modelos experimentais pequenos em laboratório, mas um modelo competitivo de vários bilhões de parâmetros demanda grandes volumes de tokens e infraestrutura multi-GPU ou de cluster. Para a etapa inicial do SAVITS, essa alternativa não apresenta uma relação custo-benefício favorável.

4. Infraestrutura computacional e tempo de processamento

A infraestrutura deve ser analisada em função da operação a ser realizada. Inferência, RAG e treinamento possuem perfis de consumo muito diferentes. Na inferência, a memória é utilizada principalmente para armazenar os pesos, o contexto e caches internos. No treinamento, além dos pesos, são necessários gradientes, ativações e estados do otimizador, o que pode multiplicar várias vezes o consumo de memória.

A CPU continua importante mesmo quando o modelo é executado em GPU. Ela participa do carregamento de dados, tokenização, preparação de batches, serviços de API, processamento documental, execução do banco vetorial e coordenação geral do sistema. Para uma estação experimental, 8 a 16 núcleos são suficientes para tarefas básicas; para

um servidor institucional que execute simultaneamente ingestão, *embeddings*, RAG e inferência, 16 a 32 núcleos oferecem uma margem mais adequada.

A RAM deve ser dimensionada para o conjunto do sistema, e não apenas para o LLM. Uma máquina com 32 GB pode atender protótipos, mas 64 GB é uma base mais confortável para RAG e ajuste eficiente. Para uma infraestrutura de pesquisa e desenvolvimento, 128 GB permite manter bases maiores em memória, executar múltiplos serviços e trabalhar com *datasets* de treinamento sem pressão constante de swap. Em cenários de *full fine-tuning* ou multi-GPU, 256 GB ou mais passam a ser recomendáveis.

A VRAM é o componente mais determinante para modelos locais. GPUs de 12 a 16 GB permitem trabalhar com modelos pequenos e com versões 7B ou 8B quantizadas. GPUs de 24 GB ampliam consideravelmente as possibilidades, especialmente para QLoRA. Uma GPU de 32 a 48 GB oferece espaço para modelos maiores, contextos mais extensos e experimentos com menor necessidade de *offloading*. Em uma infraestrutura institucional voltada à pesquisa, uma única GPU de 48 GB pode ser mais simples de operar que duas GPUs menores quando o objetivo principal é acomodar um modelo maior em um único dispositivo.

O armazenamento deve utilizar SSD NVMe sempre que possível. Modelos, checkpoints, adaptadores, *datasets*, bases documentais e índices vetoriais podem rapidamente ocupar centenas de gigabytes. Um ambiente que pretende comparar várias famílias de modelos deve considerar pelo menos 2 TB, sendo 4 TB uma recomendação mais realista para o SAVITS. Backup e preservação dos conjuntos de dados devem ser tratados separadamente, preferencialmente em infraestrutura de armazenamento institucional dedicada.

A rede possui importância limitada quando toda a solução opera em um único servidor. Uma interface de 1 GbE atende protótipos e serviços básicos, enquanto 10 GbE é recomendável para integração com servidores de dados, armazenamento e outros componentes da plataforma. Em ambientes multi-GPU dentro do mesmo servidor, PCIe e a interconexão entre GPUs são mais relevantes que a rede Ethernet. A rede se torna crítica quando o treinamento é distribuído entre servidores, cenário no qual 25, 100 GbE ou InfiniBand podem ser necessários.

Os tempos de treinamento dependem mais do número de tokens processados do que do simples número de exemplos. Um conjunto com 10 mil registros de 500 tokens possui aproximadamente 5 milhões de tokens por época; o mesmo número de registros com 4 mil tokens possui 40 milhões. Por isso, estimativas devem considerar tamanho médio das sequências, número de épocas, *batch size, gradient accumulation* e *throughput* efetivo da GPU. A Tabela 1 apresenta um sumário de todas essas questões.

Tabela 1 - Ordens de grandeza para infraestrutura e tempo.

Estratégia	**Infraestrutura típica**	**Memória recomendada**	**Tempo aproximado**
Modelo pronto / Prompt / Few-shot	1 GPU de 8-16 GB ou CPU	16-32 GB RAM	Sem treinamento; implantação em minutos
RAG	1 GPU de 8-16 GB + NVMe	32-64 GB RAM	Minutos a horas para ingestão e indexação
QLoRA 3B	1 GPU de 12-16 GB	32-64 GB RAM	Aproximadamente 1-3 horas
QLoRA 7-8B	1 GPU de 24 GB	64 GB RAM	Aproximadamente 2-8 horas
QLoRA 13-14B	1 GPU de 24-48 GB	64-128 GB RAM	Horas a cerca de 1 dia
Full fine-tuning 7-8B	48-80 GB agregados ou multi-GPU	128 GB RAM ou mais	Horas a dias
Continued pretraining	Multi-GPU	128-256 GB RAM ou mais	Dias a semanas
Treinamento do zero	Cluster	256 GB RAM a TB	Dias, semanas ou meses

Fonte: elaborada pelo autor

Os valores são indicativos e dependem do modelo, do *dataset*, da precisão numérica e da configuração de treinamento.

Uma forma prática de estimar o treinamento é utilizar a relação entre o número total de tokens processados e o *throughput* efetivo da infraestrutura. Se um experimento processar 30 milhões de tokens ao longo de todas as épocas e a configuração sustentar 1.000 tokens por segundo, o tempo bruto será de aproximadamente 30.000 segundos, ou 8,3 horas. Na prática, checkpoints, avaliação, carregamento de dados e variações de eficiência aumentam esse tempo. A fórmula é útil principalmente para comparar cenários e demonstrar por que *datasets* com o mesmo número de registros podem demandar tempos muito diferentes.

5. Análise comparativa e aderência ao SAVITS

A comparação das alternativas indica que o SAVITS não precisa escolher entre uma solução completamente estática e um modelo treinado do zero. Existe uma faixa intermediária de estratégias que oferece elevada flexibilidade com custos computacionais moderados. Modelo pré-treinado e *prompting* são apropriados como primeira etapa porque permitem construir uma referência de desempenho sem treinamento. RAG agrega conhecimento institucional e rastreabilidade sem alterar pesos. QLoRA permite especializar o comportamento quando o *prompting* não é suficiente. Essas três camadas formam uma sequência tecnológica coerente e incremental.

O treinamento completo apresenta maior liberdade de modificação, mas também maior custo e complexidade. Para justificá-lo, seria necessário demonstrar experimentalmente que QLoRA ou outras técnicas eficientes não alcançam a qualidade necessária. *Continued pretraining* exige ainda um corpus de dimensão suficiente para modificar de forma significativa a distribuição linguística do modelo. No início do projeto, a base longitudinal e os dados do piloto ainda estarão sendo construídos; portanto, é improvável que exista volume e diversidade suficientes para tornar essa alternativa prioritária.

Treinar um modelo do zero apresenta a menor aderência inicial. Embora possa produzir um ativo totalmente controlado pela instituição, o custo computacional e de dados é muito superior e o risco científico também é maior. Em vez de competir com modelos gerais já treinados em grandes corpora, é mais racional aproveitar esses modelos como base e concentrar os recursos do projeto na especialização, na integração com dados institucionais, na avaliação e na governança.

Do ponto de vista de soberania digital, é importante destacar que utilizar um modelo previamente treinado não significa necessariamente utilizar um serviço externo. Os pesos podem ser hospedados e executados localmente, preservando o controle sobre o ambiente.. A Tabela 2 resume essas condições.

Tabela 2 - Comparação qualitativa das estratégias consideradas para o SAVITS.

Estratégia	Altera pesos	Conhecimento externo	Custo	Complexidade	Aderência inicial
Modelo pronto / Prompt	Não	Somente contexto	Baixo	Baixa	Alta
Few-shot / ICL	Não	Somente contexto	Baixo	Baixa	Alta
RAG	Não	Sim	Baixo-Médio	Média	Muito alta
LoRA / QLoRA	Parcialmente	Não	Médio	Média	Alta
Full fine-tuning	Sim	Não	Alto	Alta	Média
DAPT / Continued pretraining	Sim	Corpus	Muito alto	Alta	Baixa inicialmente
Distillation	Sim	Teacher	Alto	Alta	Possível no futuro
Treinamento do zero	Totalmente	Corpus próprio	Muito alto	Muito alta	Baixa

Fonte: elaborada pelo autor

A soberania, nesse caso, está mais relacionada ao controle da execução, dos dados, das versões e da infraestrutura do que à necessidade de produzir internamente todos os parâmetros do modelo

6. Proposta de arquitetura evolutiva para o SAVITS

A recomendação é adotar uma arquitetura modular, de modo que o modelo de linguagem não seja acoplado rigidamente à plataforma. A camada de aplicação do SAVITS deve interagir com o modelo por meio de uma API interna. Essa API pode direcionar solicitações para diferentes modelos, aplicar *templates* de prompt, acionar mecanismos de recuperação e registrar informações necessárias à auditoria. A separação entre serviço de aplicação, serviço de recuperação e serviço de inferência facilita a substituição de modelos e reduz dependência tecnológica.

Na primeira fase, recomenda-se selecionar dois ou mais modelos abertos de portes diferentes e avaliá-los sem treinamento. O objetivo é estabelecer uma baseline para as

tarefas de sumarização, extração temática, classificação e resposta em linguagem natural. Nessa etapa, devem ser testados system prompts e *few-shot prompting*. O resultado permitirá identificar se um modelo menor, como 3B, é suficiente para determinadas tarefas ou se modelos na faixa de 7B a 14B apresentam ganhos relevantes.

A segunda fase deve introduzir RAG. Os documentos e dados autorizados devem ser preparados, fragmentados e indexados, mantendo metadados capazes de representar origem, data, tipo de documento, nível de acesso e outras propriedades relevantes. A recuperação não deve depender apenas de similaridade vetorial; filtros por metadados e regras de autorização devem fazer parte do pipeline. A resposta produzida pelo modelo deverá incluir referências às fontes ou, pelo menos, mecanismos que permitam ao usuário abrir os trechos utilizados.

A terceira fase deve avaliar QLoRA somente se forem observadas limitações comportamentais recorrentes. Um *dataset* supervisionado pode ser construído a partir de exemplos validados por especialistas. O objetivo não deve ser memorizar documentos, mas ensinar o modelo a executar tarefas segundo os protocolos do SAVITS. A combinação recomendada é, portanto, modelo base local, adaptador QLoRA opcional e RAG para fornecimento de conhecimento.

Em uma evolução posterior, a infraestrutura poderá incorporar um grafo de conhecimento. O grafo pode representar relações que a recuperação vetorial não expressa de forma explícita, como vínculos entre programas, instituições, indicadores, documentos, pesquisadores, públicos-alvo e resultados. O modelo generativo atuaria então como camada de interpretação e linguagem natural sobre duas fontes complementares: recuperação textual e relações estruturadas.

Essa arquitetura é compatível com a metodologia iterativa prevista no projeto. Cada camada pode ser desenvolvida e validada separadamente, reduzindo o risco de implementar desde o início uma solução excessivamente complexa. Também permite que a equipe compare objetivamente o ganho de cada componente e identifique em que ponto a complexidade adicional deixa de produzir benefício significativo.

7. Recomendação de infraestrutura e estratégia de avaliação

Para uma prova de conceito, seria possível iniciar com uma máquina de 8 a 16 núcleos de CPU, 32 a 64 GB de RAM, uma GPU de aproximadamente 16 GB de VRAM e 1 a 2 TB de armazenamento NVMe. Essa configuração permitiria executar modelos pequenos, versões quantizadas de modelos 7B ou 8B, *embeddings* e RAG. Entretanto, considerando que o projeto pretende desenvolver uma plataforma institucional e realizar pesquisa comparativa, essa configuração apresenta pouca margem para evolução.

Como infraestrutura recomendada para pesquisa e desenvolvimento, sugere-se um servidor ou *workstation* com 16 a 32 núcleos de CPU, 128 GB de RAM, uma GPU com 32 a 48 GB de VRAM, pelo menos 4 TB de armazenamento NVMe e rede de 10 GbE. Sempre que possível, o equipamento deve possuir fonte, refrigeração, slots PCIe e espaço físico que permitam uma segunda GPU no futuro. Essa configuração oferece margem para executar modelos na faixa de 3B a 14B, trabalhar com contextos maiores, gerar *embeddings*, manter serviços RAG e realizar QLoRA com maior segurança operacional.

Uma infraestrutura avançada, com 256 a 512 GB de RAM e duas a quatro GPUs de 48 a 80 GB, deve ser tratada como expansão condicionada a evidências experimentais. Ela passa a fazer sentido caso o projeto demonstre necessidade de *full fine-tuning, continued pretraining* ou maior concorrência de inferência. A vantagem de postergar essa aquisição é permitir que a demanda real seja conhecida antes de comprometer recursos com hardware de alto custo.

A escolha definitiva do modelo não deve ser feita apenas por reputação ou benchmark público. Recomenda-se construir um *benchmark* SAVITS contendo exemplos representativos das tarefas reais. Para sumarização, podem ser utilizados documentos com sínteses de referência; para extração temática, textos previamente classificados por especialistas; para consulta, perguntas cuja resposta e evidência sejam conhecidas. Todos os modelos devem ser avaliados sobre o mesmo conjunto de tarefas e nas mesmas condições.

A avaliação deve combinar qualidade e desempenho computacional. Entre as métricas de qualidade devem estar correção, relevância, cobertura, fidelidade às fontes,

qualidade da sumarização e comportamento diante de ausência de evidência. Em RAG, é importante medir separadamente a qualidade da recuperação e a qualidade da geração, porque uma resposta incorreta pode ter origem em documentos mal recuperados e não necessariamente no modelo. Entre as métricas computacionais, devem ser registrados tempo de resposta, tokens por segundo, uso máximo de VRAM, RAM e CPU, além do tamanho dos artefatos gerados.

A reprodutibilidade exige o registro de todas as configurações. Para cada experimento, a equipe deve registrar nome e versão do modelo, quantização, *tokenizer*, parâmetros de geração, prompt de sistema, exemplos *few-shot*, modelo de *embeddings*, método de *chunking*, tamanho dos *chunks*, sobreposição, número de documentos recuperados, *reranker* utilizado, versão da base e hardware empregado. Sem esse controle, alterações relativamente pequenas podem tornar os resultados incomparáveis.

8. Considerações sobre governança, segurança e conclusão

A execução local melhora o controle sobre os dados, mas não torna a solução automaticamente segura. A arquitetura deve prever autenticação, autorização, segregação de ambientes, registro de acesso e políticas claras para armazenamento de prompts e respostas. Em particular, um sistema RAG pode recuperar corretamente um documento que determinado usuário não deveria visualizar. Por isso, as regras de acesso devem ser aplicadas no momento da recuperação e não apenas depois que a resposta foi gerada.

Outro risco é o *prompt injection*. Um documento incorporado à base pode conter conteúdo que seja interpretado pelo modelo como uma instrução. O sistema deve diferenciar claramente as instruções da aplicação, a consulta do usuário e o conteúdo recuperado. Documentos devem ser tratados como dados, e não como comandos. Essa preocupação é especialmente importante em ambientes que incorporam conteúdos de múltiplas origens.

O caráter de apoio à decisão definido pelo SAVITS deve permanecer explícito na interface. Respostas de LLMs são probabilísticas e podem apresentar erros mesmo quando utilizam fontes corretas. A existência de evidência recuperada melhora a

auditabilidade, mas não garante que a interpretação gerada esteja correta. Em situações de maior impacto, o resultado deve ser revisado por pessoa qualificada e apresentado como apoio analítico, não como decisão automática.

A análise realizada indica que a primeira etapa do SAVITS não exige treinamento de um modelo próprio do zero nem *full fine-tuning* de grande escala. A combinação de modelos abertos previamente treinados, execução local, *prompt engineering* e RAG apresenta a melhor relação inicial entre custo, controle, atualização do conhecimento e rastreabilidade. QLoRA deve ser incorporado como mecanismo de especialização caso os experimentos demonstrem que *prompting* e RAG não são suficientes para garantir o comportamento esperado.

A infraestrutura recomendada deve, portanto, ser dimensionada para suportar inferência de modelos pequenos e médios, RAG, geração de *embeddings* e ajuste eficiente. Um servidor com 16 a 32 núcleos de CPU, 128 GB de RAM, GPU de 32 a 48 GB de VRAM, 4 TB de NVMe e rede de 10 GbE constitui um ponto de equilíbrio adequado entre capacidade atual e possibilidade de evolução. *Full fine-tuning, continued pretraining, distillation* e treinamento do zero devem permanecer como alternativas futuras, condicionadas à disponibilidade de dados, às necessidades demonstradas empiricamente e ao benefício esperado.

Em síntese, recomenda-se que o SAVITS não seja concebido como um sistema dependente de um único LLM, mas como uma infraestrutura modular de IA generativa. O ativo principal deve ser a arquitetura capaz de combinar modelos intercambiáveis, dados governados, mecanismos de recuperação, evidências documentais e processos de avaliação reproduzíveis. Essa opção preserva a autonomia tecnológica do projeto, reduz o risco de obsolescência e permite incorporar novos modelos e técnicas ao longo do desenvolvimento sem reconstruir toda a plataforma.

Referências

DETTMERS, T. et al. QLoRA: Efficient Finetuning of Quantized LLMs. Advances in Neural Information Processing Systems, 2023.

HU, E. J. et al. LoRA: Low-Rank Adaptation of Large Language Models. International Conference on Learning Representations, 2022.

LEWIS, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 2020.

OUYANG, L. et al. Training Language Models to Follow Instructions with Human Feedback. Advances in Neural Information Processing Systems, 2022.

RAFAILOV, R. et al. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Advances in Neural Information Processing Systems, 2023.

IBICT. SAVITS: Sistema de Avaliação e Validação de Impactos em Tecnologias Sociais. Brasília, 2026.

gov.br
Documento assinado digitalmente
FABIO DINIZ ROSSI
Data: 23/09/2026 15:48:09-0300
Verifique em https://validar.iti.gov.br

Fábio Diniz Rossi

De acordo:

gov.br
Documento assinado digitalmente
WASHINGTON LUIS RIBEIRO DE CARVALHO SEG
Data: 28/09/2026 17:16:50-0300
Verifique em https://validar.iti.gov.br

Washington Luís Ribeiro de Carvalho Segundo
Coordenador do projeto