Sonar Municipal: Recomendação de Políticas Públicas a partir de Projetos de Lei Municipais e Indicadores Oficiais

Thiago Ambiel
University of São Paulo - USP
São Carlos-SP, Brazil
thiago.ambiel@usp.br

André C. P. L. F. de Carvalho
University of São Paulo - USP
São Carlos-SP, Brazil
andre@icmc.usp.br

Abstract

Municipal governments often need to draft, compare, and justify legislative proposals under limited time and resources. This report describes a reproducible method to discover Brazilian SAPL instances, extract municipal Bills (Projetos de Lei), transform their short summaries (ementas) into action-oriented recommendations, and retrieve relevant actions via semantic search. We also describe how time-series indicators from official sources can be used to approximate real-world effects, and how similar proposals can be grouped into policy clusters to improve statistical robustness. Finally, we present a web platform that operationalizes the method for non-technical users. Keywords: municipal legislation; SAPL; semantic search; text-to-text models; open data; reproducibility

Resumo

Gestores municipais frequentemente precisam redigir, comparar e justificar proposições legislativas com restrição de tempo e de recursos. Este relatório descreve um método reprodutível para (i) descobrir instâncias do SAPL no Brasil, (ii) extrair Projetos de Lei (PLs), (iii) transformar ementas em recomendações de ação, e (iv) recuperar ações relevantes por busca semântica. Também descrevemos como indicadores em séries temporais de fontes oficiais podem ser usados para estimar efeitos no mundo real, e como agrupar projetos similares em "políticas públicas" aumenta o poder estatístico e a explicabilidade. Por fim, apresentamos uma plataforma web que torna o método acessível a usuários sem perfil técnico. Palavras-chave: legislação municipal; SAPL; busca semântica; modelos texto-para-texto; dados abertos; reprodutibilidade

1 Introdução

A produção legislativa municipal no Brasil ocorre em um cenário de alta diversidade institucional. Mesmo quando municípios enfrentam problemas semelhantes (por exemplo, segurança pública ou evasão escolar), as soluções propostas variam em linguagem, escopo e grau de detalhamento. Essa heterogeneidade dificulta comparar, reutilizar e avaliar propostas existentes, reduzindo a eficiência da tomada de decisão baseada em evidências.

Este trabalho descreve o projeto Sonar Municipal (plataforma web: `https://sonar-municipal.vercel.app/`), cujo objetivo é apoiar a elaboração e a análise de Projetos de Lei (PLs) por meio de:

- descoberta e coleta sistemática de PLs em instâncias do SAPL;
- transformação de ementas em recomendações de ação;
- busca semântica de ações a partir de perguntas do usuário;
- simulação aproximada de efeitos em indicadores oficiais ao longo do tempo;
- agrupamento de PLs similares em "políticas públicas" para análise conjunta.

Contribuições. As principais contribuições são: (C1) método auditável para descoberta de instâncias do SAPL e extração exaustiva de PLs via API; (C2) transformação de ementas em recomendações de ação; (C3) busca semântica baseada em embeddings; (C4) análise de indicadores com agrupamento de projetos similares; (C5) plataforma web para usuários não técnicos.

2 Fundamentação e conceitos utilizados

2.1 SAPL e padronização do processo legislativo

O SAPL (Sistema de Apoio ao Processo Legislativo) é mantido pelo Programa Interlegis e amplamente adotado por casas legislativas brasileiras para informatizar e dar publicidade ao processo legislativo [1]. O sistema organiza etapas como proposição, tramitação e votação, favorecendo padronização. Por ser de código aberto [2], muitas instâncias compartilham estruturas de dados e padrões de consulta e, em vários casos, expõem uma API.

2.2 APIs HTTP, REST, OpenAPI e Swagger

Uma API (Interface de Programação de Aplicações) é um conjunto de regras que permite que programas acessem funcionalidades e dados de um serviço. Neste trabalho, consideramos APIs acessadas via HTTP/HTTPS.

O estilo REST (Representational State Transfer) organiza a API em recursos acessados por URLs, com operações padronizadas (por exemplo, GET para consulta) [3]. Um endpoint é a URL associada a um recurso específico (por exemplo, uma lista de matérias).

OpenAPI descreve formalmente endpoints, parâmetros e respostas em um formato legível por máquina [4]. O Swagger UI renderiza essa descrição em uma interface navegável para exploração e teste [5].

2.3 Paginação

APIs costumam retornar coleções de forma paginada, dividindo resultados em páginas para reduzir carga e tamanho da resposta. Para uma extração exaustiva, é necessário iterar todas as páginas até o esgotamento dos resultados [6].

2.4 Embeddings, busca semântica e banco vetorial

Embedding é um vetor numérico que representa um texto de modo que conteúdos semanticamente próximos fiquem próximos no espaço vetorial. Busca semântica recupera itens por similaridade entre embeddings, e não por coincidência literal de termos.

Um banco vetorial armazena vetores e permite consultas eficientes por similaridade. Neste projeto, utilizamos o Qdrant [7, 8].

2.5 Proveniência e FAIR

Proveniência é informação sobre origem e transformações do dado, útil para auditoria, reprodutibilidade e reuso. O modelo PROV-DM do W3C formaliza esse conceito [9]. Também buscamos aderência prática aos princípios FAIR (Encontrável, Acessível, Interoperável e Reutilizável) [10].

3 Metodologia

3.1 Visão geral do pipeline

A Figura 1 resume o pipeline em seis etapas: (E1) municípios, (E2) descoberta de instâncias do SAPL, (E3) extração de PLs, (E4) transformação de ementa em ação, (E5) indexação vetorial e (E6) indicadores e agrupamento.

3.2 Descoberta de instâncias do SAPL e extração de Projetos de Lei

3.2.1 Universo de busca e cobertura territorial

O quadro amostral (lista de municípios) foi obtido via API de localidades do IBGE [11]. Essa escolha reduz viés de seleção por listas manuais e permite replicação a partir da mesma fonte.

3.2.2 Geração de candidatos e validação por evidência pública

A descoberta foi implementada como geração de candidatos + validação. A geração usa heurísticas (regras práticas baseadas em padrões observáveis de URL) para criar endereços prováveis de portais SAPL. A validação busca evidências públicas de compatibilidade (respostas e páginas de consulta) antes de aceitar a instância.

Algorithm 1 Descoberta e validação de instâncias SAPL (visão conceitual)

Entrada: Lista de municípios M (IBGE)
Saída: Lista de instâncias SAPL validadas S

$S \leftarrow \emptyset$
for cada município $m \in M$ **do**
  $C \leftarrow$ gerar candidatos de URL para m (heurísticas)
  for cada candidato $c \in C$ **do**
    if rota pública de consulta responde e conteúdo é compatível com SAPL **then**
      adicionar c em S
      break ▷ evita duplicidade para o mesmo município
    end if
  end for
end for
return S

3.2.3 Estrutura conceitual da API do SAPL

Quando exposta, a API tende a seguir REST [3], com endpoints de coleção (listagem) e item (detalhe). Em várias instâncias, a descrição OpenAPI e a interface Swagger UI podem estar disponíveis [4, 5].

3.2.4 Extração de Projetos de Lei

A extração foi desenhada para ser adaptável ao vocabulário local, já que rótulos variam entre casas legislativas. O método consulta o catálogo de tipos de matéria, identifica quais rótulos correspondem a "Projeto de Lei" e percorre todas as páginas até o esgotamento (paginação) [6].

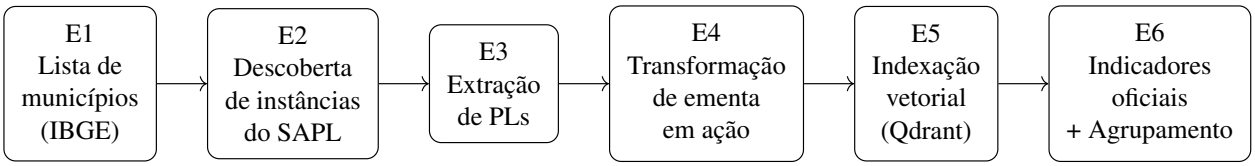

Figura 1: Pipeline conceitual do Sonar Municipal.

Algorithm 2 Extração exaustiva de PLs via API (visão conceitual)

Entrada: Instância SAPL validada s
Saída: Conjunto de PLs P extraídos de s

$P \leftarrow \emptyset$
$T \leftarrow$ consultar catálogo de tipos de matéria em s
$T_{pl} \leftarrow$ selecionar tipos que correspondem a "Projeto de Lei"
for cada tipo $t \in T_{pl}$ **do**
    $page \leftarrow 1$
    while há resultados na página $page$ **do**
        $R \leftarrow$ requisitar lista paginada de matérias do tipo t na página $page$
        $P \leftarrow P \cup R$
        $page \leftarrow page + 1$
    end while
end for
return P

3.2.5 Resultado da primeira execução e customizações locais

Na primeira execução dos coletores, foram encontradas 1.259 instâncias do SAPL. A extração automatizada de PLs foi possível em 322 instâncias, totalizando 220.065 PLs coletados. A discrepância decorre de customizações locais (mudanças de rotas, proxies, autenticação, regras anti-robô e parametrizações) que impedem ou dificultam o acesso programático.

3.2.6 Rastreabilidade

Para auditoria, cada registro mantém proveniência mínima: instância de origem, identificadores do processo legislativo e carimbo temporal de coleta. O PROV-DM formaliza esse conceito [9].

3.3 Construção do tradutor de ementas para ações

Ementas têm alta variabilidade e linguagem abreviada. Para reduzir ruído na busca e melhorar a interpretação do resultado, o projeto transforma ementas em recomendações de ação, preservando a semântica.

Foi criado um conjunto de dados de pares *ementa* $\rightarrow$ *ação* com 1.000 amostras sintéticas, geradas por um LLM da família GPT-5.1 [12, 13]. O objetivo é realizar transferência de aprendizado, isto é, aproveitar um modelo já treinado e ajustá-lo à tarefa com poucos exemplos. Como modelo base para o fine-tuning, utilizamos o PTT5-v2, voltado ao português [14]. O PTT5-v2 é uma variante do modelo T5 do Google, treinada com textos em português pela Unicamp [15].

O modelo tradutor foi treinado como tarefa seq2seq (sequência para sequência). O ajuste foi feito com QLoRA-4bit [16]. A qualidade foi avaliada por BERTScore [17], obtendo 84% na avaliação interna do projeto.

Após o treinamento, todas as ementas coletadas foram convertidas em ações, formando um conjunto finito de ações recomendáveis para indexação.

3.4 Codificação das ações para busca semântica

Para indexar ações e consultas no mesmo espaço vetorial, foi utilizado o Multilingual E5 [18], adequado ao cenário de pergunta e resposta. As embeddings foram armazenadas no Qdrant [7, 8], permitindo recuperar as top-K ações mais similares a uma pergunta.

Para viabilizar hospedagem de baixo custo do serviço de embeddings, uma opção prática é usar Text Embeddings Inference (TEI) [19].

3.5 Indicadores para simulação de efeito no mundo real

Indicadores em séries temporais foram construídos por município, usando fontes oficiais. No indicador educacional, por exemplo, utilizam-se microdados do Censo Escolar (INEP) [20]. A comparação do indicador na data de apresentação do PL com valores posteriores produz uma estimativa operacional de variação, com horizonte escolhido pelo usuário.

Observação: essa estimativa é associativa, não causal. Ela descreve mudanças no período posterior, mas não prova que o PL causou a variação.

3.6 Agrupamento de projetos em políticas públicas

Para aumentar robustez, PLs semanticamente similares são agrupados em clusters chamados "políticas públicas". A análise conjunta permite calcular efeito médio e dispersão. Para ranquear políticas, foi definida a métrica:

$$Q = \frac{n_{\text{positivos}}}{n} \times \frac{n}{n+1}$$

onde $n_{\text{positivos}}$ é o número de municípios com efeito considerado positivo (de acordo com o objetivo do indicador) e n é o total de municípios no grupo. O fator $\frac{n}{n+1}$ reduz o peso de grupos muito pequenos.

3.7 Desenvolvimento da plataforma web

A plataforma foi desenvolvida em React com Next.js e hospedada na Vercel [21]. A arquitetura integra três componentes: interface web, serviço de embeddings e banco vetorial (Qdrant), conectados por APIs.

Funcionalidades principais:

1. recomendação de políticas públicas com estimativa de efeito esperado;
2. detalhamento de PLs (ementa, data, link do SAPL e gráfico do indicador);
3. busca semântica no acervo para apoiar ideação e comparação de propostas.

4 Resultados

Os resultados desta etapa concentram-se na cobertura da coleta e na qualidade do tradutor de ementas.

Tabela 1: Resumo da primeira execução de descoberta e extração.

Métrica	Valor
Instâncias do SAPL encontradas	1.259
Instâncias do SAPL com extração bem-sucedida	322
Projetos de Lei coletados (total)	220.065

Como mostra a Tabela 1, a primeira execução dos coletores atingiu escala nacional, com extração bem-sucedida em parte das instâncias descobertas. Na avaliação interna, o tradutor ementa → ação atingiu BERTScore de 84%, indicando preservação semântica adequada para uso prático.

5 Limitações e ameaças à validade

Este estudo enfrenta limitações operacionais e metodológicas. Do ponto de vista operacional, a disponibilidade das instâncias do SAPL é variável: há instabilidades, mudanças de URL e barreiras como proxies, autenticação e regras anti-robô, o que reduz a cobertura efetiva da coleta. Além disso, customizações locais e diferenças de configuração afetam a padronização dos endpoints e dificultam a extração automática. No aspecto semântico, a variação de rótulos e tipos de proposição entre casas legislativas pode levar a omissões ou inconsistências na identificação de Projetos de Lei. Já na análise de impacto, a variação de indicadores ao longo do tempo não pode ser atribuída de forma causal a um único PL, pois outras políticas e fatores contextuais podem influenciar os resultados. Por fim, o uso de um conjunto de dados sintético para treinar o tradutor pode introduzir vieses do gerador e limitar a generalização do modelo em cenários reais.

6 Conclusão

O Sonar Municipal integra coleta sistemática de PLs via SAPL, transformação de ementas em ações, busca semântica por embeddings, análise com indicadores e agrupamento de políticas públicas. O projeto entrega uma ferramenta que aproxima evidências legislativas e dados oficiais do fluxo de trabalho de gestores públicos, com foco em reprodutibilidade e auditabilidade. Como resultado, a plataforma favorece a comparação entre propostas, a identificação de alternativas e a tomada de decisão informada, mantendo transparência sobre a origem e o processamento dos dados.

Referências

[1] Senado Federal e Programa Interlegis. *Sistema de Apoio ao Processo Legislativo (SAPL)*. URL: https://www12.senado.leg.br/interlegis/produtos/sapl (acesso em 07/01/2026).

[2] Interlegis. *SAPL: Sistema de Apoio ao Processo Legislativo (repositório oficial)*. URL: https://github.com/interlegis/sapl (acesso em 07/01/2026).

[3] Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures". Tese de dout. University of California, Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (acesso em 07/01/2026).

[4] OpenAPI Initiative. *OpenAPI Specification (v3.2.0)*. 2024. URL: https://spec.openapis.org/oas/v3.2.0.html (acesso em 07/01/2026).

[5] Swagger. *Swagger UI: REST API Documentation Tool*. URL: https://swagger.io/tools/swagger-ui/ (acesso em 07/01/2026).

[6] Django REST Framework. *Pagination*. URL: https://www.django-rest-framework.org/api-guide/pagination/ (acesso em 07/01/2026).

[7] Qdrant. *Qdrant Documentation*. URL: https://qdrant.tech/documentation/ (acesso em 07/01/2026).

[8] Qdrant. *Collections (conceitos)*. URL: https://qdrant.tech/documentation/concepts/collections/ (acesso em 07/01/2026).

[9] W3C. *PROV-DM: The PROV Data Model*. 2013. URL: https://www.w3.org/TR/prov-dm/ (acesso em 07/01/2026).

[10] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg et al. "The FAIR Guiding Principles for scientific data management and stewardship". Em: *Scientific Data* 3 (2016), p. 160018. DOI: 10.1038/sdata.2016.18.

[11] IBGE. *API de Localidades (documentação)*. URL: https://servicodados.ibge.gov.br/api/docs/localidades (acesso em 07/01/2026).

[12] OpenAI. *GPT-5.1 Model (OpenAI API)*. 2026. URL: https://platform.openai.com/docs/models/gpt-5.1 (acesso em 07/01/2026).

[13] OpenAI. *GPT-5.1 Instant and GPT-5.1 Thinking: System Card Addendum*. 2025. URL: https://openai.com/index/gpt-5-system-card-addendum-gpt-5-1/ (acesso em 07/01/2026).

[14] Marcos Piau, Matheus P. Vieira et al. "PTT5-v2: A Closer Look at Continued Pretraining of T5 Models for the Portuguese Language". Em: *arXiv* (2024). URL: https://arxiv.org/abs/2406.10806 (acesso em 07/01/2026).

[15] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". Em: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: https://jmlr.org/papers/v21/20-074.html (acesso em 07/01/2026).

[16] Tim Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs". Em: *arXiv* (2023). URL: https://arxiv.org/abs/2305.14314 (acesso em 07/01/2026).

[17] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". Em: *arXiv* (2019). URL: https://arxiv.org/abs/1904.09675 (acesso em 07/01/2026).

[18] Liang Wang et al. "Multilingual E5 Text Embeddings: A Technical Report". Em: *arXiv* (2024). URL: https://arxiv.org/abs/2402.05672 (acesso em 07/01/2026).

[19] Hugging Face. *Text Embeddings Inference (TEI)*. URL: https://huggingface.github.io/text-embeddings-inference/ (acesso em 07/01/2026).

[20] INEP. *Censo Escolar: Microdados (dados abertos)*. URL: https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/censo-escolar (acesso em 07/01/2026).

[21] Vercel. *Next.js on Vercel (documentação)*. URL: https://vercel.com/docs/frameworks/full-stack/nextjs (acesso em 07/01/2026).