NEURALMIND INTELIGÊNCIA ARTIFICAL

TERRANOVA CONSULTORIA

Proposta de Pesquisa e Desenvolvimento Tecnológico:

INA^2: Um Modelo Fundacional de **IN**strução **A**ssistida por **IN**teligência **A**rtificial para o Tribunal de Contas da União (TCU)

CAMPINAS

20 de maio de 2022

Resumo

Em atendimento ao Edital de Chamamento Público para Encomenda Tecnológica (ETEC) de Instrução Assistida por Inteligência Artificial (IA) do Tribunal de Contas da União (TCU), o consórcio formado pelas empresas NeuralMind e Terranova Consultoria submete esta **proposta de serviços de pesquisa e desenvolvimento de um modelo funcional (unificado) de INstrução Assistida por INteligência Artificial: a INA²**. A INA² atenderá a todos os objetivos listados pelo TCU no edital de contratação e **posicionará o Tribunal na fronteira tecnológica no uso de inteligência artificial para controle externo e prestação jurisdicional em matéria orçamentária e financeira.**

O elemento central da proposta consiste na **construção e manutenção de um modelo único (e proprietário), do TCU, de aprendizado profundo generativo, similar ao GPT-3, com capacidade de aprendizado *few-shot***, que será capaz de cumprir as tarefas de processamento de linguagem natural (NLP) com baixo custo de treinamento, ampla generalização e plena flexibilidade na execução de tarefas do Tribunal. A INA² **alimentará dados não-estruturados ao Painel de Jurimetria, expandindo a interpretação e sugestões de encaminhamento de processos para além do que seria possível apenas com metadados processuais**. Por fim, esta solução permitirá que as equipes de tecnologia da informação, de fiscalização e de julgamento trabalhem de forma unificada para a consecução dos objetivos do Tribunal e que a transferência de *know-how* seja facilmente assimilada pela equipe, já que esta proposta unifica as soluções das diversas tarefas sob um framework único.

O desenvolvimento tecnológico desta solução ocorrerá num prazo de até 30 (trinta) meses, a contar da data de assinatura do contrato de encomenda tecnológica entre o consórcio e o TCU. O consórcio antevê duração de **18 (dezoito) meses para a etapa um**, detecção e extração de informações de peças processuais, **12 (doze) meses para a etapa dois**, painel de jurimetria, e **12 (doze) meses para a etapa três**, redação de peças processuais. Em virtude das intersecções entre estes períodos, ainda restariam 12 (doze) meses, reservados para o marco de sustentação e evolução desta solução para absorção pelo TCU. **O investimento necessário para o seu desenvolvimento totaliza R$ 6.186.600,00**, conforme cronograma físico financeiro constante desta proposta.

(página intencionalmente deixada em branco)

1. Introdução

Os termos aprendizado de máquina (ou aprendizado estatístico), inferência preditiva e inteligência artificial (IA) apenas recentemente passaram a integrar o vocabulário de negócios no mundo todo, ainda que seu desenvolvimento tenha começado na década de 1950 (Solomonoff, 1985).[1] Atualmente, estas tecnologias despertam o interesse do setor público, de organizações sem fins lucrativos e de pesquisadores que desejam alinhar sua expertise em uma área do conhecimento com novas tecnologias e análises de dados para resolver problemas desafiadores. Neste contexto, **o consórcio formado pela NeuralMind e a Terranova Consultoria, e representado pela NeuralMind, submete**, neste documento, proposta para o desenvolvimento de módulo de instrução assistida do Tribunal de Contas da União (TCU).

O modelo aqui proposto, o qual denominamos INA[2] (INstrução Assistida por INteligência Artificial), é um modelo generativo de aprendizado profundo (treinado de forma auto-supervisionada), que consistirá em submódulos capazes de realizar:

4. **Extração e organização de informações não-estruturadas** de textos por meio de processamento de linguagem natural (NLP);
5. **Análise processual descritiva e preditiva** com base nas informações não-estruturadas e os metadados processuais;
6. **Sugestão de ações (encaminhamento)** com base nas informações textuais, análises descritivas e preditivas;
7. **Redação dos documentos processuais** mais relevantes para o TCU.

Atualmente, grande parte dos sistemas de NLP comerciais utiliza modelos customizados para cada tipo de tarefa. Um sistema extrator de informações de um processo é comumente composto por modelos de reconhecimento de entidades nomeadas (NER) (para extrair o nome do denunciante, por exemplo) e modelos classificadores (para predizer o tipo do processo, por exemplo). **O desenvolvimento desses modelos é demorado e custoso pois requer dados de treinamento específicos para cada tarefa**. A cada nova tarefa (por exemplo, a extração de novas informações), serão repetidas as atividades de coleta de dados, anotação, treinamento e avaliação do modelo preditivo. Este retrabalho é um problema comum na indústria e também se manifesta

[1] Solomonoff, R. J. (1985). "The Time Scale of Artificial Intelligence: Reflections on Social Effects". *Human Systems Management*, 5(2), 149-153.

em modelos de NLP modernos, aqueles que usam redes neurais pré-treinadas (como o BERT). Do ponto de vista financeiro, esta rotina aumenta significativamente os custos do produto pois cada rodada exige proporcionalmente mais horas de trabalho de engenheiros(as) de software, cientistas de dados e especialistas do Direito, uma vez que a capacidade preditiva dos modelos decresce conforme se aumenta a complexidade da tarefa a ser resolvida.[2]

O consórcio NeuralMind-Terranova, portanto, não acredita que esta abordagem seja adequada para solucionar as tarefas conforme descritas pelo TCU no edital de encomenda tecnológica; com modelos clássicos, ao final do projeto, o Tribunal arcaria com tarefas financeiramente e computacionalmente custosas; os ajustes dos modelos complexos exigiriam conhecimentos avançados de NLP ou aprendizado de máquina da equipe técnica do TCU; e, finalmente, a solução não seria facilmente escalável para outros tipos ou assuntos processuais. **Em vista disso, propõe-se o desenvolvimento de modelo unificado (INA^2) de aprendizado *few-shot* que dispensa treinamento de vários modelos especializados e que soluciona os problemas de classificação, extração e geração de texto definidas nas etapas 1 e 3 da encomenda tecnológica**. A INA^2 reduziria a dependência de uma grande coleção de documentos anotados para treinamento (sendo necessários apenas documentos anotados para validação), facilitaria a adaptação para a realização das tarefas por meio de engenharia de *prompts* e poderia ser utilizada, de imediato, em diversos assuntos processuais do TCU.

Esta abordagem vem sendo adotada pelos líderes da indústria de inteligência artificial. Uma equipe de pesquisadores do Google recentemente propôs a utilização de um único modelo treinado em centenas ou milhares de tarefas.[3] Esta visão também é compartilhada por organizações que fazem pesquisa de ponta, como a OpenAI, DeepMind e Meta, que cada vez mais publicam resultados de modelos multilíngues ou multimodais (i.e., modelos que tratam texto, imagem e voz) capazes de realizar diversas tarefas simultaneamente.[4] **O principal obstáculo, contudo, para a implementação desta solução é a experiência em pré-treinamento e transferência de aprendizado de modelos de linguagem. É justamente este o diferencial do nosso consórcio: desde 2017, a NeuralMind treina e disponibiliza modelos de linguagem**

[2] Em outras palavras, um poder preditivo de 80% em tarefas mais complexas leva proporcionalmente mais tempo para se obter do que o mesmo poder preditivo em tarefas mais simples.
[3] https://blog.google/technology/ai/introducing-pathways-next-generation-ai-architecture/
[4] https://arxiv.org/abs/2106.13884; https://arxiv.org/pdf/2103.03206.pdf; https://openai.com/blog/clip/

***open-source* em português para a comunidade lusófona de ciência e tecnologia**. O BERTimbau, **modelo pré-treinado pela NeuralMind**, é o modelo em português mais baixado da HuggingFace (com mais de 1 milhão de downloads) e serviu de base para treinamento de modelos específicos em diferentes domínios (financeiro, biomedicina e Direito); **e, desde 2014, a Terranova usa dados estruturados do Direito para fazer análise preditiva e painéis de jurimetria para contrapartes no setor público, setor privado e terceiro setor.**

A INA^2 será construída tanto de forma programática, via API, para uso especializado das equipes técnicas do Tribunal, **como também em plataforma web**, de modo que usuários menos especializados possam fornecer dados, consumir análises e gerar documentos processuais. As saídas da INA^2 serão ingeridas pelo Painel de Jurimetria, a ser desenvolvido como produto da etapa 2, que reunirá indicadores processuais com base nas informações textuais extraídas pelo modelo e em metadados processuais. **A INA^2 concentrará, portanto, em apenas um local, todas as tarefas desenvolvidas ao longo do projeto.**

Para demonstrar o poder desta solução, **apresentamos uma extração feita por um modelo generativo em português com aprendizado *few-shot* (que já usamos internamente em projetos nossos) a partir de trecho dos Termos de Referência (TdR) deste Edital de Encomenda Tecnológica.** A Figura 1 contém um trecho da petição inicial usada como exemplo na página 65 do TdR concatenada com a pergunta: “Quais são as irregularidades apontadas no trecho acima?”. O texto em negrito é usado como entrada para o modelo e a saída produzida é o texto que não está em negrito.

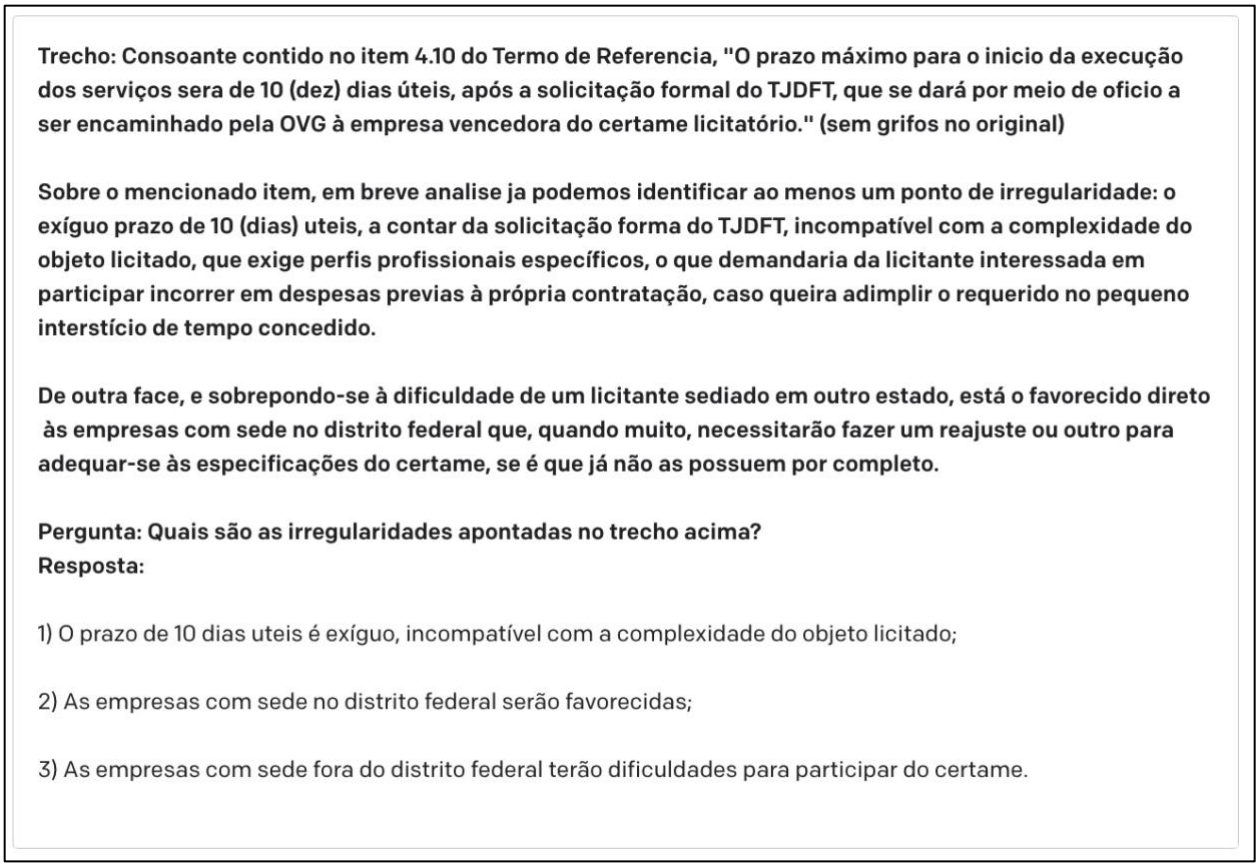

Figura 1: Exemplo de uso de modelo generativo para extrair informações não-estruturadas de documentos usando aprendizado *few-shot*.

O modelo responde com as principais alegações do peticionário — exíguo prazo (resposta 1) e favorecimento a empresas do Distrito Federal (respostas 2 e 3) — **sem nenhum exemplo de treinamento**. Este seria o ponto de partida da solução proposta pelo consórcio NeuralMind-Terranova, cuja metodologia, implementação e adequação ao propósito do TCU estão apresentados nesta proposta.

2. Justificativa

As soluções propostas pelo consórcio NeuralMind-Terranova **são superiores aos modelos clássicos porque contemplam não somente o estado da arte em inteligência artificial, mas também o modelo de negócio e fluxo processual do TCU.** Entendemos que, por exemplo, o módulo de instrução assistida deve ser escalável tanto no que diz respeito a tipo de processo quanto a assunto processual. Uma vez que há heterogeneidade na disponibilização e na anotação de peças processuais, **não acreditamos que seja adequado adotar solução que dependa prioritariamente da anotação de documentos para treinamento de modelos de aprendizado de máquina.** Além disso, entendemos que a instrução de processos e propostas de encaminhamento devem ser fundamentadas em análises de dados estruturados e não-estruturados, já que, por princípio, a solução de controvérsias por via judicial exige análise de aderência do Direito ao fato concreto; isto é, não se pode encaixotar, automatizar ou substituir o processo decisório de funcionários do TCU por algoritmos de inteligência artificial. Estes servem, na verdade, de subsídio às atividades do Tribunal.

Propomos construir um modelo de linguagem unificado próprio do TCU, similar ao GPT-3 da OpenAI. Este modelo foi um dos primeiros a apresentar capacidades de aprendizado *in-prompt*, no qual se consegue realizar a tarefa desejada usando apenas alguns exemplos de treino (*few-shot*) fornecidos como entrada em tempo de inferência. O modelo é apenas pré-treinado na tarefa de modelagem da linguagem (i.e., predição da próxima palavra, dada as anteriores como entrada) e não precisa de *fine-tuning* na tarefa específica. Como dispensa treinamento, podemos usar o mesmo modelo para realizar diversas tarefas, bastando mudar o *prompt* que usamos como entrada para o modelo.[5]

Do ponto de vista de infraestrutura computacional, usar um modelo único para servir diversas tarefas traz muitas simplificações. Por exemplo, podemos usar apenas um servidor com grande poder computacional que será compartilhado por todos os estágios do fluxo de extração de informações e redação de peças. Se tivéssemos um modelo para cada tarefa, possivelmente precisaríamos de diversos servidores, cada um servindo um ou dois desses modelos. Além disso, devido à complexidade das tarefas do TCU, o número de subtarefas intermediárias é grande

[5] Ao longo desta proposta utilizamos o termo modelo few-shot para nos referir a um modelo que tem a capacidade executar uma tarefa usando apenas poucos exemplos do prompt, i.e., não há ajuste dos pesos do modelo depois de pré-treinado na tarefa de modelagem da linguagem.

(provavelmente da ordem de centenas); portanto, vemos que é financeiramente inviável treinar modelos para cada uma delas, mesmo que os modelos necessitem de apenas algumas centenas de exemplos de treinamento. Soma-se a isso o fato de que as tarefas podem ser modificadas e adicionadas frequentemente, o que exigiria novas coletas de dados de treino e ajuste de hiperparâmetros se usássemos modelos específicos.

O uso de modelo único e criação de tarefas e subtarefas a partir de *prompts* simplifica a transferência de *know-how* para os funcionários do TCU, pois não exige deles experiência em coleta de dados e treinamento e validação de redes neurais profundas. A manutenção e aprimoramento das tarefas e subtarefas poderão ser feitas pelos funcionários do TCU de forma muito mais simples se comparada com métodos tradicionais de fine-tuning de modelos de linguagem. Por exemplo, se for notado que uma tarefa apresenta baixo desempenho para um novo tipo de documento recebido, a equipe do TCU precisará apenas elaborar e testar novos prompts até que o desempenho na tarefa seja satisfatório. Na abordagem tradicional, teria que ser feita nova coleta de dados e retreino dos modelos.

Finalmente**, esta proposta aponta para o futuro do uso de IA em linguagem natural**. Os avanços recentes na área começaram com modelos BERT e T5, que exigem dados anonimizados anotados manualmente e fine-tuning de modelos para cada tarefa e subtarefa. Hoje, **as soluções estão migrando para modelos únicos com centenas de bilhões de parâmetros com capacidade de aprendizado *few-shot***. Os *hardwares* de treinamento desses modelos estão avançando exponencialmente e dentro de 4 a 5 anos serão mais acessíveis do que hoje.[6,7]

[6] A GPU NVIDIA H100 anunciada, em março de 2022, promete desempenho 6 a 9 vezes maior que a A100.
[7] A escolha, portanto, da alternativa que treinaria vários modelos em tarefas especializadas colocaria o Tribunal mais em risco de superação de tecnologia.

3. Objetivos

O objetivo geral deste projeto é **desenvolver tecnologia unificada de inteligência artificial** que contribua, em todos os três ciclos de desenvolvimento, para:

1. Detectar e extrair informações de peças processuais.
2. Propor encaminhamento de demandas feitas ao TCU (via painel de jurimetria).
3. Redigir documentos da coletânea jurisprudencial e publicações do TCU.

Conjuntamente, esta **tecnologia tornará a administração judicial e prestação jurisdicional do TCU mais eficiente, uniforme e responsiva.** Esta tecnologia será construída utilizando uma combinação de ferramentas *open-source* (em linguagem de programação Python, JavaScript, R e suas respectivas APIs) e ferramentas customizadas, desenvolvidas exclusivamente para o TCU (como o modelo de linguagem unificado aqui proposto). Ressalta-se que esta **tecnologia será disponibilizada tanto por meios programáticos**, isto é, APIs nas linguagens de programação comumente utilizadas pela equipe de tecnologia da informação do TCU, **como também em plataformas *online*** que consumirão dados e modelos em produção e retornarão as informações de interesse às diversas equipes do TCU.

A seguir, estão listados os objetivos específicos deste projeto. Estes objetivos estão separados pelos marcos conforme descritos no edital de encomenda tecnológica, uma vez que estes marcos se repetem ao longo de cada um dos três ciclos.

3.1. Detecção e Extração de Informações Processuais (Marco 1)

O objetivo principal da etapa de detecção e extração de informações de peças processuais é **facilitar a submissão e acelerar a análise de processos** de competência do TCU. Embora o Tribunal tenha adotado solução uniforme de submissão de petições iniciais, supõe-se que a submissão de petição própria ainda seja mecanismo frequentemente utilizado pelos autores de processos submetidos ao Tribunal, uma vez que esta é a prática comum dos operadores do Direito nos diversos órgãos do Poder Judiciário. Uso de documentos padronizados, experiência no exercício do direito e viés de *status quo*[8] são alguns dos motivos pelos quais os peticionários manteriam sua prática mesmo diante da existência de sistema unificado. **A inversão da lógica de**

[8] Kahneman, Daniel, Jack L. Knetsch, and Richard H. Thaler. 1991. "Anomalies: The Endowment Effect, Loss Aversion, and Status Quo Bias." *Journal of Economic Perspectives*, 5 (1): 193-206.

recebimento de processos, conforme sugerida no edital de encomenda tecnológica, contudo, **facilitaria o preenchimento do formulário por peticionários cuja preferência é a submissão de documento próprio sem prejuízo aos peticionários que já fazem uso do formulário padrão**. Para medir o sucesso deste objetivo, propõe-se comparar os tempos de submissão de processos, o tempo de análise e a qualidade dos pleitos antes e depois da adoção do sistema proposto pela NeuralMind-Terranova (com testes A/B, por exemplo).[9]

3.2. Painel de Jurimetria (Marco 2)

Com o Painel de Jurimetria, objetiva-se **fundamentar o encaminhamento de demandas em dados objetivos e baseados em jurisprudência consolidada no TCU**. Para além do fundamento normativo ou processual, o desenvolvimento de pesquisa quantitativa do Direito fornece base empírica para decisões de órgãos que desempenham função jurisdicional. **As matérias sobre as quais o TCU tem competência**, isto é, disputas orçamentárias e financeiras, **são candidatas clássicas à análise de impacto econômico e à mensuração de custos e benefícios financeiros de um determinado encaminhamento.** Em comum, estes cálculos dependem não só da mensuração do impacto da medida corretiva tomada pelo tribunal, mas também da probabilidade de que estas medidas sejam aplicadas corretamente e produzam efeitos no caso concreto. Para tanto, faz-se necessário o uso de modelos de inferência preditiva baseados em dados históricos estruturados (metadados processuais) e dados não-estruturados (argumentos em peças processuais, por exemplo). Assim, **os auditores e ministros do Tribunal poderão saber os efeitos probabilísticos de suas decisões em cada caso.** A avaliação deste objetivo também pode ser feita de forma experimental; contudo, aqui, a alternativa mais adequada seria a aplicação de um questionário junto aos funcionários do TCU sobre sua avaliação da utilidade do Painel de Jurimetria para instruir ou decidir sobre os processos que lhes são designados.[10]

[9] Testes A/B são técnicas experimentais, comuns em equipes de experiência de usuário (UX), que medem o ganho de usar um sistema B em comparação a um sistema A. Os peticionários seriam aleatoriamente alocados para submeter pleito via sistema atual e sistema novo. Ao final das submissões, se poderia comparar tempo de submissão e qualidade das peças de grupos de tratamento e controle para medir o resultado da adoção do novo sistema. Processo idêntico seria realizado com os auditores do TCU, cujos tempos de análise e instrução em cada sistema poderiam nos indicar o ganho de se adotar novo fluxo de peticionamento.

[10] Sniderman, Paul M. Some Advances in the Design of Survey Experiments. *Annual Review of Political Science*, 2018, 21:1, 259-275.

3.3. Redação de Peças Processuais, Instruções e Comunicados (Marco 3)

Finalmente, o objetivo final deste projeto é **aumentar a produtividade dos funcionários do TCU** ao reduzir o tempo que gastam em tarefas rotineiras e mecânicas (como as de comunicação aos interessados). **Este objetivo baseia-se em evidência empírica** já consensual na literatura econômica[11], qual seja, **de que a incorporação de novas tecnologias transforma tarefas de uma profissão e privilegia tarefas analíticas, cognitivas e criativas** (justamente as atividades mais importantes e difíceis dentro de órgãos com função jurisdicional, como o TCU) **em detrimento de tarefas mecânicas e rotineiras**. Portanto, a tecnologia desenvolvida neste projeto pode potencialmente aumentar a produtividade do trabalho e agregar mais valor às funções típicas dos funcionários do TCU. Para mensurar a consecução deste objetivo, propomos medir a qualidade deste marco através da acurácia do modelo proposto.

[11] Acemoglu, D., e Autor, D. (2011). Skills, Tasks and Technologies: Implications for Employment and Earnings. In: *Handbook of Labor Economics*. Editor: Orley Ashenfelter and David Card.

4. Metodologia

O consórcio proponente entende que a melhor forma de se atacar o problema da extração e estruturação de informações de peças processuais, recomendações de encaminhamento e redação de comunicados é **por meio da reestruturação das tarefas e ciclos propostos no edital de chamamento público.** Portanto, ao invés de propor a replicação dos marcos 1, 2 e 3 em cada um dos ciclos de desenvolvimento tecnológico, propõe-se, nesta seção, que **o projeto seja dividido por marcos e que todos os ciclos sejam objeto de análise em cada marco.** A metodologia aqui proposta pode ser aplicada para diferentes tipos de processos (representações, denúncias ou fiscalizações, por exemplo) e com ampla cobertura de assuntos processuais (aquisições públicas, Lei 14.133/2021 ou demais assuntos). Nas subseções a seguir, está descrita a metodologia proposta para desenvolver soluções em cada ciclo do projeto.

4.1. Detecção e Extração de Informações Processuais

O marco 1 deste projeto requer a extração e predição de diversas informações que podem estar dispersas em peças processuais, principalmente na petição inicial e seus anexos. Dentre os desafios desta etapa, aponta-se para: 1) a diversidade de tarefas a serem realizadas para extrair as informações; 2) a dispersão de informações relevantes em documentos processuais diferentes; 3) a heterogeneidade das peças processuais própria do exercício do Direito e das diversas estratégias argumentativas.

Nossa visão para execução destas tarefas é simples e pragmática: **propomos a utilização de um único modelo de linguagem (INA^2) pré-treinado em diversos documentos — incluindo textos jurídicos (do TCU e de outros órgãos da Justiça) — e que tenha capacidades de aprendizado *few-shot***, i.e., que tenha a capacidade de executar uma tarefa usando apenas alguns exemplos na entrada e que não necessite de *fine-tuning* em tarefas específicas. Um exemplo popular de modelo com aprendizado *few-shot* é o GPT-3. A proposta de um modelo unificado como o GPT-3 é simples: a partir de um modelo pré-treinado, a INA^2 será capaz de prever a palavra seguinte com base nas palavras anteriores sem necessidade de treinamento (i.e., ajuste dos pesos do modelo) nas tarefas de extração de informação.

Na Figura 2, ilustramos o funcionamento de um modelo *few-shot* já utilizado internamente com um exemplo de extração de nome e endereço de requerentes em um processo de reintegração de imóvel. O texto em negrito é chamado de *prompt* e é fornecido como entrada ao modelo. O exemplo de extração é o "Exemplo 1", que contém a entrada (o texto alvo da extração) e a saída esperada (nome e endereço). O "Exemplo 2" é a entrada cuja extração é necessária. **Os *tokens* da resposta (destacados em azul) são produzidos um a um de maneira auto-recursiva, isto é, o token produzido no passo anterior é concatenado ao *prompt*, que é então usado como entrada para o modelo para geração do próximo *token*.**

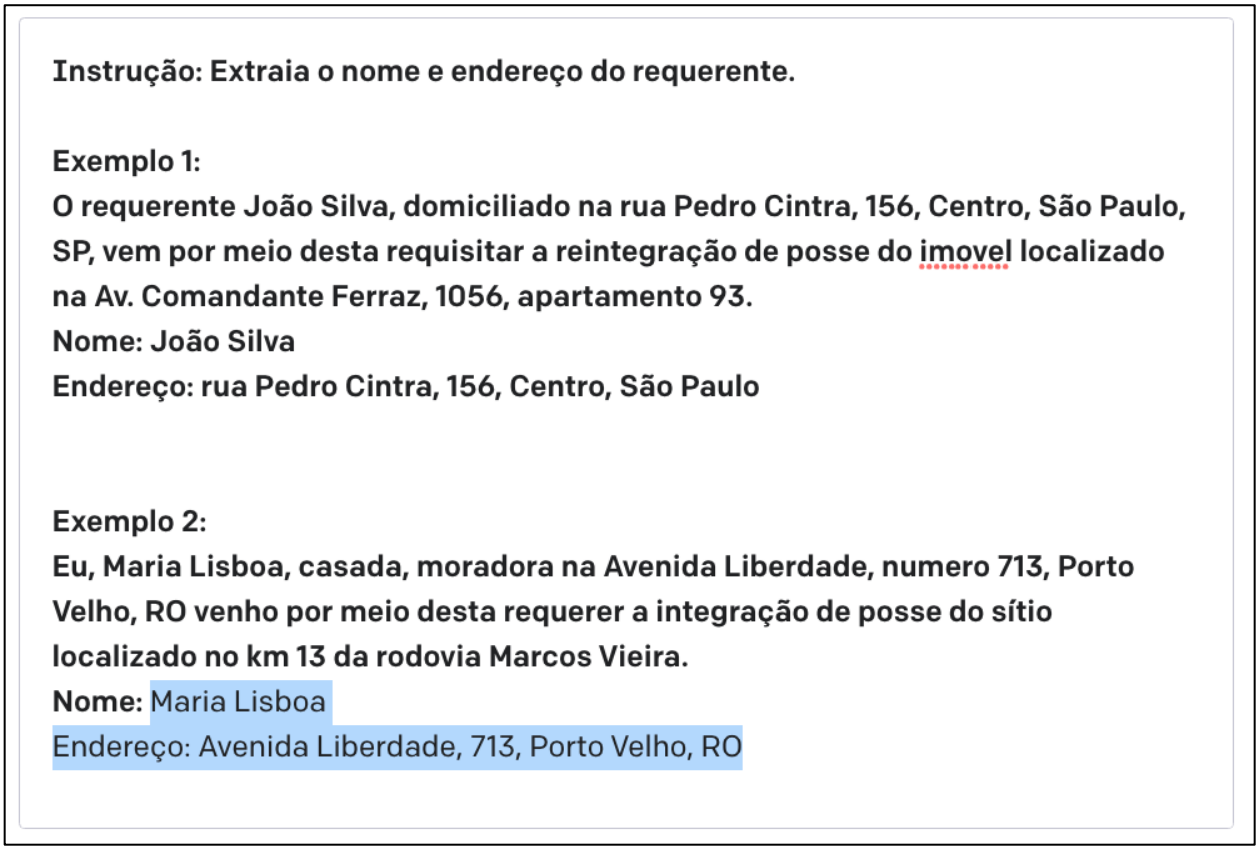

Figura 2: Extração de informações usando um modelo *few-shot* com uma amostra de treinamento na entrada (exemplo 1) que exemplifica a tarefa a ser realizada.

Com apenas uma breve descrição da tarefa e um exemplo de entrada, o modelo foi capaz de extrair corretamente o nome e o endereço do segundo exemplo. Note que o modelo não se confundiu com nomes como "Marcos Vieira" e omitiu a palavra "número" ao gerar o endereço, formatando a saída conforme o exemplo fornecido. **Esse processo de extração serve para exemplificar a metodologia que usaremos para extrair para todas as informações que devem**

ser retiradas das peças processuais, quais sejam: as perguntas constantes do guia de avaliação do exame de admissibilidade, o objeto do processo, o resumo das alegações do autor, as irregularidades, a existência de perigo na demora etc.[12]

As vantagens de se usar um modelo *few-shot* neste projeto são: 1) ele **dispensa a necessidade de treinamento para tarefas específicas**, o que geralmente requereria centenas ou milhares de exemplos para um treinamento estável; 2) **dispensa a necessidade de servir e manter vários modelos em produção**; 3) caso haja alguma modificação nas tarefas desempenhadas pelo TCU, conseguiremos **adicionar ou remover passos do fluxo de análise rapidamente[13], bastando confeccionar novos *prompts*** e especificar como as saídas serão utilizadas. Criar-se-á um repositório de *prompts* para facilitar a adaptação da solução ao longo do projeto e depois em sua eventual adoção.[14]

Pré-treinamento no domínio: No primeiro ano do projeto, iremos utilizar modelos *few-shot* disponíveis via *APIs* comerciais, como o GPT-3 da OpenAI e T0++, disponível na HuggingFace. A NeuralMind, empresa líder do consórcio, já utiliza estes modelos há meses em projetos internos. **Os resultados mostram que, apesar de pré-treinados em documentos da web, esses modelos têm bom desempenho em documentos do domínio jurídico em português.** A adoção de um modelo pré-treinado como este também permite reduzir os custos de desenvolvimento nesta fase de prova de conceito da solução tecnológica. **Para demonstrar a robustez desta proposta, as predições dos exemplos que são apresentados nesta proposta foram obtidas a partir de um destes modelos já utilizados pela nossa equipe.**

Diversos artigos recentes em NLP, contudo, mostram que o pré-treinamento destes modelos na língua e domínio no qual serão utilizados aumenta a acurácia das predições. [redacted] demonstram o ganho de acurácia em modelos como o BERTimbau[15] e o PTT5, disponibilizados publicamente pela NeuralMind. **A partir dos modelos existentes disponíveis**

[12] Apresentamos um exemplo simplificado na Figura 2 pela maior facilidade de se demonstrar o poder da solução proposta.
[13] Uma vez que esta solução não exige conhecimento de programação de computadores ou de modelos de inteligência artificial.
[14] A inspiração para esta metodologia vem de um trabalho recente de um time de pesquisa de ponta da HuggingFace e pode ser encontrada nos links a seguir: https://github.com/bigscience-workshop/promptsource; https://arxiv.org/pdf/2202.01279.pdf.
[15] Souza, F., Nogueira, R., & Lotufo, R. (2020, October). BERTimbau: pretrained BERT models for Brazilian Portuguese. In Brazilian Conference on Intelligent Systems (pp. 403-417). Springer, Cham.

publicamente, como o GPT-J[16] e T0++[17], iremos realizar mais uma rodada de pré-treinamento usando documentos jurídicos em português, fornecidos pelo TCU, para que o modelo unificado resultante deste projeto de desenvolvimento tecnológico seja ajustado para as tarefas do TCU. Neste pré-treino, não precisaremos de tantos dados quanto no pré-treino original (que foi da ordem de 100 bilhões de palavras), pois partiremos dos pesos (checkpoint) do modelo existente já pré-treinado em textos da web, que incluem textos em português. Antevemos que este treinamento será feito apenas no segundo ano do projeto, após confirmarmos que nossa metodologia funciona com os modelos *few-shot* existentes. **O consórcio NeuralMind-Terranova está unicamente posicionado para entregar essa solução ao TCU, já que temos ampla experiência em fazer pré-treinamento de modelos.** Conforme discutido na introdução desta proposta, o BERTimbau é o modelo em português com mais downloads no site da HuggingFace e é usado por diversos órgãos governamentais, empresas e instituições de pesquisa ao redor do mundo.

Pipeline low-code: Uma das características importantes do projeto será o uso de *pipelines low-code*, que usam um único modelo de aprendizado *few-shot* para diversas tarefas sem necessidade de treinamento para cada uma delas. Para cada tarefa, apenas alguns (por exemplo, três) pares de entrada e saída desejada são usados como entrada para o modelo na forma de texto. Ilustramos na figura abaixo a visão geral deste *pipeline*.

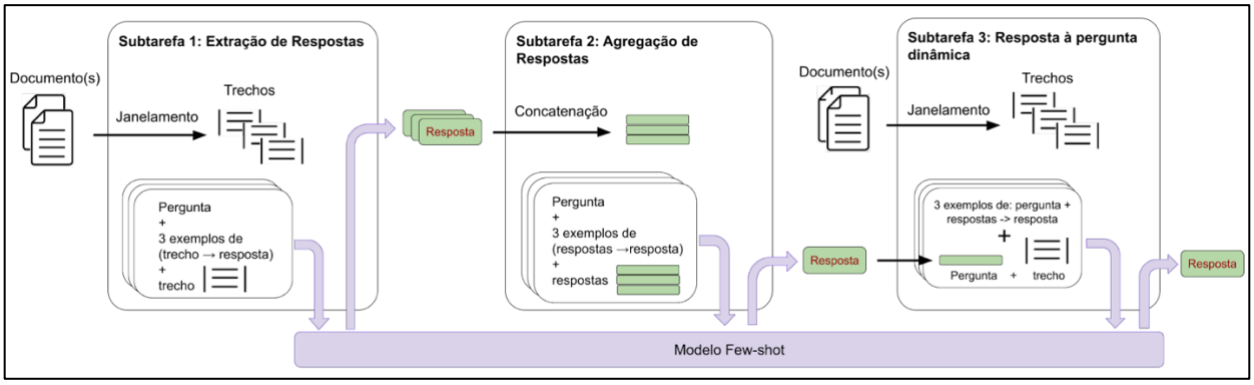

Figura 3: Visão geral do *pipeline few-shot* proposto.

O sistema ilustrado possui três subtarefas que, conjuntamente, executam uma tarefa de responder uma pergunta sobre um conjunto de documentos. O número de subtarefas não é

[16] https://huggingface.co/EleutherAI/gpt-j-6B.
[17] https://huggingface.co/bigscience/T0pp.

rígido e pode ser ajustado conforme a necessidade da tarefa principal. Neste exemplo, **a primeira subtarefa consiste em extrair respostas de um documento.** Estes documentos são geralmente longos e, portanto, primeiramente divididos em trechos menores compostos por sentenças consecutivas para que possam ser consumidos pelo modelo *few-shot*. Chamamos essa etapa de janelamento. Cada janela é um trecho d_i do documento principal, onde i indexa o número de janelas do documento. A entrada para o modelo é formada pela pergunta q, que serve como uma espécie de instrução da subtarefa que deve ser realizada, N exemplos da forma (d^*, a^*), em que a^* é resposta desejada para a pergunta q sobre o trecho d^*, e o trecho d_i, cuja resposta é desejada. A concatenação dos exemplos é utilizada como entrada para o modelo, que produz uma resposta a_i por trecho. Note que os N exemplos são criados manualmente e servem como um conjunto de "treinamento". Colocamos treinamento entre aspas pois o modelo não tem seus pesos ajustados nestes exemplos; eles servem para indicar ao modelo, em tempo de inferência, a saída esperada dado um trecho.

A segunda subtarefa consiste em agrupar as respostas produzidas na primeira subtarefa em uma única resposta. Esta agregação poderia ser feita de forma simples: por exemplo, usar expressões regulares com voto da maioria para chegar a esta única resposta seria uma alternativa. Entretanto, este trabalho seria muito demorado e requereria código especializado para cada subtarefa. O modelo *few-shot* é uma solução mais robusta para resolver este problema: dada a concatenação de respostas à pergunta gerada na primeira subtarefa, condicionamos o modelo, através dos N exemplos de um outro prefixo t, a produzir uma única resposta à pergunta, assim constituindo a segunda subtarefa na figura acima.

Por último, a terceira subtarefa tem o objetivo de responder a uma pergunta dinâmica que é formada a partir da resposta da segunda subtarefa. Por exemplo, se desejamos extrair o endereço de todos os requerentes, a segunda subtarefa poderia retornar o nome dos requerentes e a terceira subtarefa teria como objetivo responder à pergunta "*Qual o endereço do {requerente}?*", onde *{requerente}* é substituído pelo nome encontrado pela segunda subtarefa.

Com esta metodologia, **resolvemos um problema recorrente quanto ao uso de modelos pré-treinados baseados na arquitetura *Transformer*: a impossibilidade de alimentá-lo com documentos longos devido ao custo quadrático de memória e computação com relação ao tamanho do texto de entrada.** Por exemplo, os maiores textos suportados por esses modelos

variam de 2048 a 4096 *tokens*.[18] Um processo do TCU, por exemplo, pode conter milhares de palavras, o que impossibilitaria fazermos predições diretas sobre estes processos sem antes extrairmos os segmentos do texto que são relevantes à tarefa. Esse é geralmente uma tarefa intermediária difícil, pois requer a anotação de dados, treino e *deploy* de modelos específicos para cada tipo de documento. Aqui, contudo, usamos três subtarefas para extrair informações de um conjunto de documentos que de outra forma seria impossível de ser consumido de uma única vez pelo modelo.

4.1.1. Exemplo 01: Extração de Indícios

Ilustramos, abaixo, este pipeline usando um exemplo de como iremos realizar a tarefa de responder à pergunta: *"A petição inicial encontra-se acompanhada de indício concernente à irregularidade ou ilegalidade apontada pelo autor?"*

[18] De maneira simplificada, um token pode ser considerado como uma palavra.

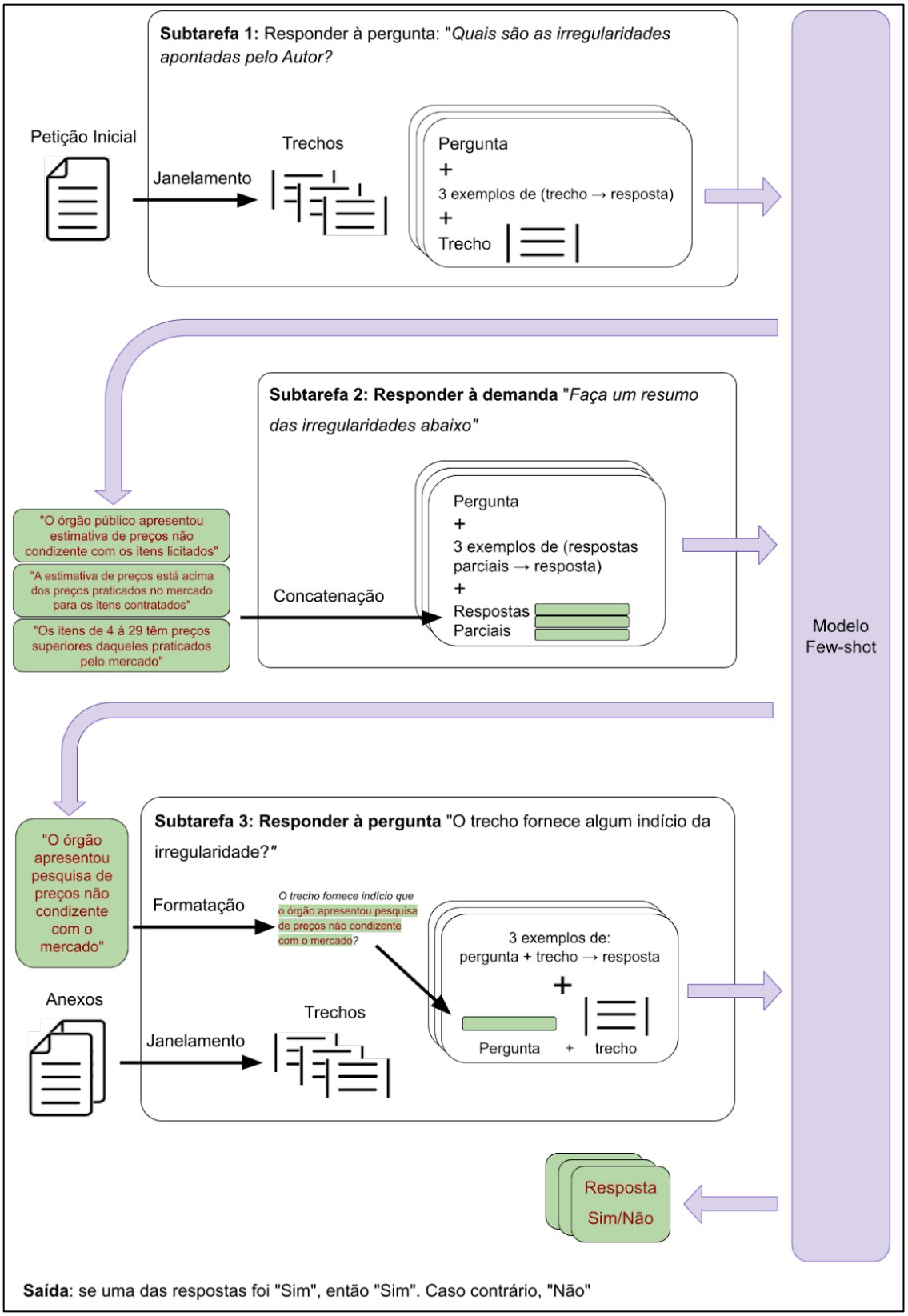

Figura 4: *Pipeline* para a tarefa de extração de indícios de irregularidade em petição inicial.

Para ilustrar a facilidade de implementação do pipeline proposto, mostramos seu código em linguagem Python:

```python
def sub_tarefa_1(petição_inicial):
    prompt = get_prompt("sub_tarefa_1")
    trechos = janelamento(petição_inicial)
    respostas = []
    for trecho in trechos:
        input = prompt.replace("{contexto}", trecho)
        resposta = modelo_few_shot(input)
        respostas.append(resposta)
    return respostas


def sub_tarefa_2(respostas):
    prompt = get_prompt("sub_tarefa_2")
    respostas = '\n'.join(respostas)  # Converte lista para string.
    input = prompt.replace("{contexto}", respostas)
    resposta = modelo_few_shot(input)
    return resposta


def sub_tarefa_3(anexos, irregularidade):
    prompt = get_prompt("sub_tarefa_3")
    trechos = janelamento(anexos)
    respostas = []
    for trecho in trechos:
        input = prompt.replace("{irregularidade}", irregularidade)
        input = input.replace("{contexto}", trecho)
        resposta = modelo_few_shot(input)
        if resposta == 'Sim':
            return True  # Qualquer trecho que comprove a irregularidade, retorne True.

    return False  # Se nenhum trecho foi encontrado, retorne False.



if __name__ == '__main__':
    petição_inicial = get_petição_inicial()  # Retorna uma string contendo o texto da petição inicial.
    anexos = get_anexos()  # Retorna uma string contendo o texto dos anexos concatenados.


    respostas = sub_tarefa_1(petição_inicial)
    irregularidade = sub_tarefa_2(respostas)
    resposta_final = sub_tarefa_3(anexos, irregularidade)


    print("A petição inicial encontra-se acompanhada de indício concernente à irregularidade ou ilegalidade apontada pelo autor?")
    print(resposta_final)
```

Figura 5: Código necessário para extração de indícios de irregularidade em petição inicial.

No código acima, **a primeira etapa consiste em converter os documentos da petição inicial e anexos de PDF para textos (prevista como atividade nesta proposta).** Alguns desses arquivos necessitarão de pré-processamentos extras. Por exemplo, documentos que contenham tabelas ou gráficos serão removidos. A razão é que estes tipos de dados estruturados ainda são um desafio para modelos baseados em redes neurais modernas. Ou seja, textos estruturados presentes em documentos anexados poderão causar uma diminuição na qualidade das predições caso presentes.

Uma vez feita a conversão de texto dos arquivos PDFs, chamamos cada uma das três subtarefas sequencialmente. Note que elas são concisas, com apenas alguns pré-processamentos nas variáveis de entrada e resposta do modelo. As chamadas `get_prompt("prompt_name")` retornam o *prompt*

manualmente projetado para cada subtarefa. **É na elaboração destes *prompts* que boa parte do esforço de implementação será gasto.** Como o *prompt* é em linguagem natural, podemos usar especialistas que não possuem conhecimentos de programação, aumentando assim o número de pessoas que podem contribuir com o projeto.

Para efeitos de verificação da corretude do sistema, iremos retornar ao usuário os pares pergunta/trecho cuja resposta foi "Sim". Isso torna o sistema mais interpretável, facilitando a correção de eventuais erros de predição. **Enfatizamos que o principal esforço de desenvolvimento nesta metodologia consiste em dividir cada tarefa** (ex: "A representação/denúncia trata de matéria de competência do TCU?") **em uma sequência de subtarefas, cada uma requerendo apenas alguns poucos exemplos e uma descrição sucinta da subtarefa.**[19]

4.1.2. Exemplo 02: Perigo na Demora

A seguir fornecemos mais um exemplo de uso do *pipeline* cuja tarefa consideramos complexa: a de responder à pergunta "*Existe perigo na demora na resposta ao pedido de medida cautelar?*"

Para respondê-la, iremos decompô-la em várias perguntas mais simples, que serão respondidas através de múltiplas subtarefas. As perguntas mais simples formam uma espécie de "checklist": caso a resposta para uma delas seja positiva, então a resposta final é positiva. Este *pipeline* é ilustrado na figura abaixo:

[19] No código acima não mostramos as funções de conversão dos documentos PDF para *string* e nem a de obtenção dos prompts para cada subtarefa. O objetivo é mostrar que é possível resolver uma tarefa complexa em menos de 50 linhas de código usando a metodologia proposta.

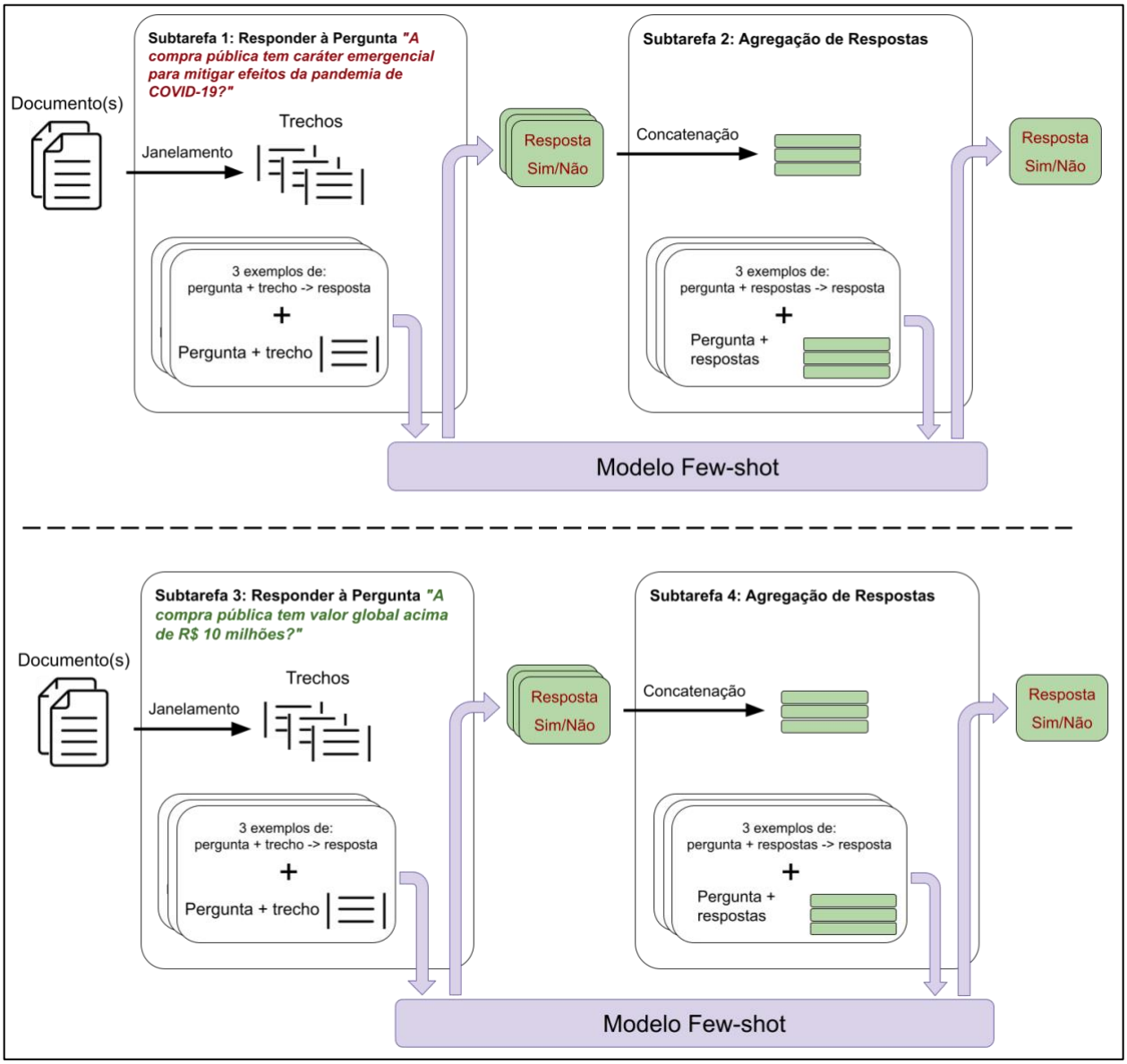

Figura 6: Tarefa de verificação de perigo na demora ao pedido de medida cautelar.

Neste exemplo, a pergunta "*Existe perigo na demora na resposta ao pedido de medida cautelar?*" foi decomposta em duas perguntas mais simples, que podem ser as mesmas usadas como guia pelos especialistas:

- *"A compra pública tem caráter emergencial para mitigar efeitos da pandemia de COVID-19?"*
- *"A compra pública tem valor global acima de R$ 10 milhões?*

Cada uma delas é respondida através de duas subtarefas: resposta por trecho (subtarefas 1 e 3) e posterior agregação das respostas em uma única resposta por documento (subtarefas 2 e 4). Neste exemplo,

se a resposta final for “Sim” para pelo menos uma das perguntas, a resposta à pergunta “*Existe perigo na demora na resposta ao pedido de medida cautelar?*” também será “Sim”.

Para efeitos de verificação da corretude do sistema, podemos retornar ao usuário os pares <pergunta, trecho> cuja resposta foi “Sim”. Isso torna o sistema mais interpretável, facilitando a correção de eventuais erros de predição.

4.1.3. Exemplo 03: Tipo de Irregularidade

Algumas subtarefas, devido à diversidade de saídas desejadas, requerem prefixos compostos por muitos exemplos. Um exemplo são tarefas de classificação com muitas classes, como o preenchimento do campo “Tipo de Irregularidade” do exame de admissão. Por exemplo, se existirem 20 classes, o prefixo deveria ser composto por, no mínimo, 20 exemplos, um para cada classe, o que resultaria em um prefixo possivelmente maior que o limite de tokens de entrada do modelo, que é tipicamente de 2048 ou 4096 tokens. **Para solucionar este problema, utilizaremos *prompts* dinâmicos**,[20] cujos exemplos do prefixo são selecionados com base no texto a ser predito. Mais especificamente, dados vários exemplos de treino (por exemplo, 100) formados por pares de <texto de entrada, texto de saída>, selecionaremos os *K* (por exemplo, três) exemplos cujo texto de entrada é mais similar ao texto de entrada a ser predito. Para calcular esta similaridade iremos utilizar o sistema de busca descrito a seguir. Ilustramos, abaixo, um exemplo de utilização de *prompts* dinâmicos para resolução da tarefa “Tipo de Irregularidade”.

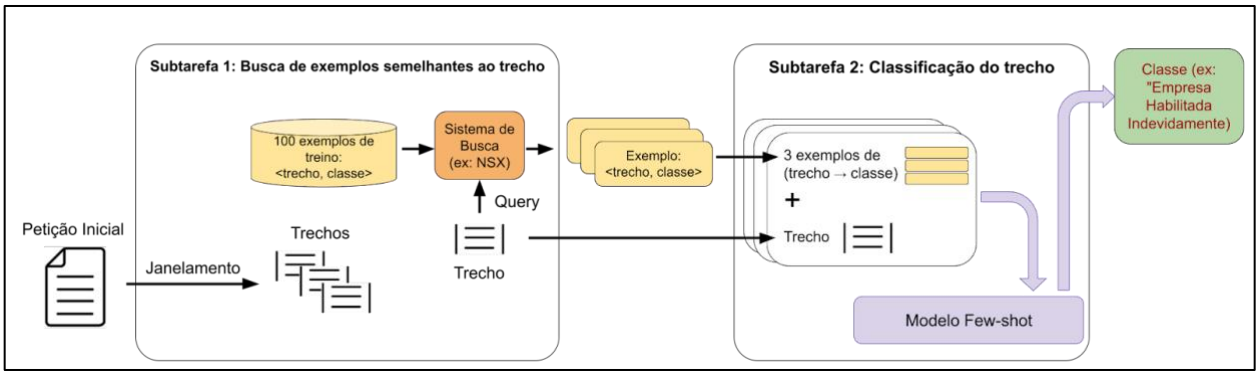

Figura 7: Exemplo de Pipeline para o Preenchimento do Campo “Tipo de Irregularidade”.

[20] Gao, T., Fisch, A., & Chen, D. (2020). Making Pre-trained Language Models Better Few-shot Learners. arXiv preprint arXiv:2012.15723.

A tarefa é composta de duas subtarefas. A primeira subtarefa consiste em encontrar exemplos similares ao trecho a ser classificado. Para tanto utilizaremos um sistema de busca moderno, como o NeuralSearchX (NSX), desenvolvido pela NeuralMind. A consulta (*query*) será um dos trechos da petição inicial, obtida após o janelamento. O sistema de busca utilizará o algoritmo BM25 e modelos ranqueadores baseados em modelos de linguagem pré-treinados (como por exemplo, Nogueira and Cho, 2019) para encontrar exemplos que sejam similares ao texto de entrada. Esta arquitetura é o estado da arte em diversos *leaderboards* de ranqueamento de documentos (como o MS MARCO) e competições organizadas pelo TREC, na qual **a NeuralMind ficou em primeiro lugar em várias trilhas entre 2019 e 2020**. Este sistema também é utilizado com sucesso na tarefa de análise de jurisprudência (CO LIEE 2021), na qual fomos campeões na tarefa 2 e segundo colocados na tarefa 1.

Neste exemplo, retornamos $K = 3$ exemplos mais similares ao trecho de uma base de treino composta por 100 exemplos de pares <trecho, classe>. Estes exemplos são então prefixados ao *prompt* da subtarefa 2, cujo objetivo será responder à pergunta "*Qual o tipo de irregularidade apontada no trecho?*". O *prompt* é enviado ao modelo *few-shot*, que retorna um texto que representa a classe. Note que a única forma do modelo saber quais são as classes possíveis é através dos exemplos prefixados no *prompt*. Ao final temos uma classe por trecho, e a classe que for a mais comum dentre os trechos será a classe da petição inicial.

Em nossos experimentos ainda não publicados, **o *prompt* dinâmico melhora bastante a qualidade das predições pois permite que uma base de treino maior seja usada.** Devido à complexidade das tarefas deste projeto e à necessidade do uso de contextos longos, antecipamos que *prompts* dinâmicos serão de grande utilidade. O *know-how* do NeuralSearchX será transferido para o TCU.

4.1.4. Reconhecimento de Assinatura

Algumas subtarefas consistem em verificar se o documento já foi assinado. Nestes casos, iremos utilizar redes neurais convolucionais treinadas para detectar assinaturas dada uma imagem do documento, i.e., sem ainda passar pelo reconhecimento ótico de caracteres (OCR). Temos vasta experiência nesta tarefa pois um dos produtos da NeuralMind (NDocs®) requer esta detecção.

4.1.5. Avaliação de Subtarefas

Apesar da metodologia proposta dispensar o treinamento de modelos, e, portanto, a necessidade de criar grandes *datasets* de treino, ainda precisamos avaliar a qualidade das predições em cada subtarefa. Para tanto, **será necessária a criação de *datasets* de *avaliação*, um para cada subtarefa.** Apesar de requererem o trabalho de especialistas, pois precisam ser de alta qualidade, **esses conjuntos de dados podem ser bem menores que os de treino comumente usados por modelos clássicos de aprendizado de máquina.** Uma vez criados, a escolha dos melhores *prompts* de cada subtarefa será feita com base no desempenho nestes *datasets* de avaliação. Cada subtarefa terá sua métrica específica. Por exemplo, em subtarefas de classificação, utilizaremos a acurácia balanceada por classe. Para tarefas de extração de informações não ambíguas, como nome do requerente, utilizaremos o *exact match*, isto é, verificamos se o texto predito e o texto desejado são exatamente iguais. Para tarefas cujo texto de saída pode ser expresso de diversas formas (ex: *Qual a irregularidade apontada pelo autor?*) utilizaremos métricas de sumarização (ROUGE), de tradução (BLEU) e de perguntas e respostas, como o F1 do saco de palavras (*bag-of-words*) da interseção das palavras preditas vs. desejadas.

A avaliação nas subtarefas será a atividade que requererá maior esforço e, consequentemente, será a que gastaremos a maior parte dos recursos. **Isto se contrapõe ao paradigma de desenvolvimento de sistemas clássicos**, no qual a maior parte do tempo se gasta na criação de regras de extração (ex: expressões regulares) ou combinando *features* para treino de classificadores simples (ex: SVM). Também é diferente do desenvolvimento de sistemas de aprendizado de máquina modernos, onde a criação de dados de treinamento e ajuste de hiperparâmetros consome recursos consideráveis. **A vantagem de nossa metodologia é a criação rápida de sub-sistemas que podem ser avaliados pelos usuários finais logo no início do projeto.**

4.1.6. Avaliação Fim-a-fim

Ao atingirmos um desempenho satisfatório em cada uma das subtarefas, avaliaremos o pipeline como um todo para medir a taxa de acerto na tarefa. Para tarefas cujas saídas são estruturadas (ex: problemas de classificação ou de extração de informações não ambíguas), criaremos um *dataset* de avaliação que tem como entrada as petições iniciais e seus anexos e a saída é a resposta desejada, isto é, a classe ou o texto exato esperado. Para tarefas cujas saídas são

em texto livre (ex: do campo “relato dos fatos”) faremos uma avaliação manual utilizando especialistas, cuja resposta será uma única nota referente a acurácia do texto gerado pelo modelo. Os resultados atingidos nesta avaliação fim-a-fim serão utilizados para o cálculo da remuneração extra variável proposta mais abaixo.

4.2. Painel de Jurimetria

O painel de jurimetria terá como objetivo principal auxiliar na etapa de instrução dos processos **ao fundamentá-la em cálculo probabilístico das ações a serem tomadas pelos funcionários do TCU**. Nessa etapa, o servidor deve produzir um relatório contendo uma análise técnica do tema em discussão para que o(a) ministro(a) relator(a) possa tomar uma decisão de qualidade.

Por conta do volume e complexidade das ações, **é necessário aprimorar o trabalho dos profissionais do TCU em duas frentes**. **A primeira é aumentar a rapidez de leitura e geração dos documentos**, a partir de atividades respectivamente desenvolvidas nos marcos 1 e 3. **A segunda é priorizar corretamente os casos a serem analisados**, de forma que os casos mais urgentes e de maior relevância sejam analisados primeiro. Este é o problema que será trabalhado no marco 2.

Diante disto, se propõe que o módulo de jurimetria tenha **duas abas principais, chamadas Panorama Geral** e **Risco.** O panorama geral tem como objetivo apresentar estatísticas do histórico de casos. No painel, será possível acessar dados de casos encerrados para a análise de estatísticas, como taxa de procedência, taxa de admissibilidade, taxa de aplicação de medidas cautelares e tempo de análise; estas análises poderão ser segmentadas por meio da aplicação de filtros e agrupamentos de processos por contexto (por exemplo, por tipo ou por assunto). A tela abaixo mostra o conceito do panorama geral. O conteúdo é meramente ilustrativo, baseado em um projeto já realizado pela Terranova Consultoria.

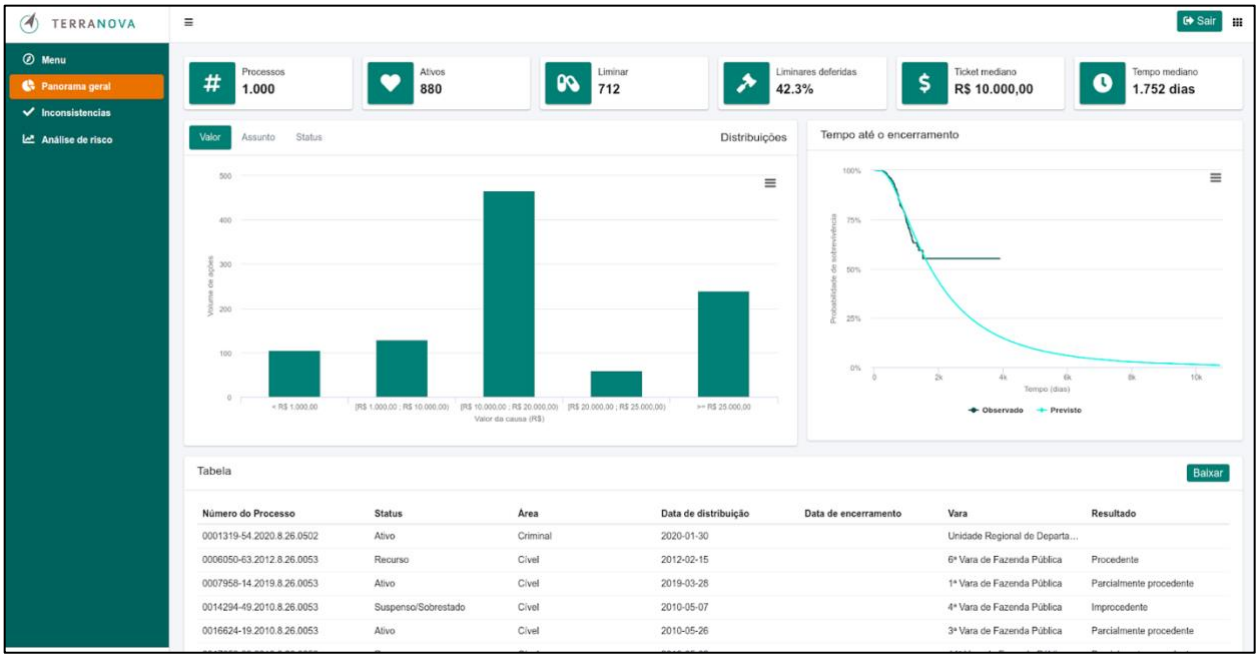

Figura 8: Aba "Panorama Geral" do painel de jurimetria.

Opcionalmente, o painel geral também poderá conter estimativas de entrada e saída de processos. A entrada de novos casos apresenta sazonalidade e **a análise das séries de tempo permitirá à equipe do TCU se preparar para épocas de maior ou menor demanda**. A Figura 9, abaixo, mostra um exemplo de visualizações de entradas, saídas e estoque de processos (quantidade de casos ativos), baseada em projeto anterior já desenvolvido pela Terranova. Para realizar as predições, serão utilizados modelos adequados de séries temporais.[21] O painel possibilita o *download* de um relatório técnico atualizado periodicamente com informações sobre a qualidade das predições, comparando diferentes competidores (suavização exponencial, ARIMA, *Gradient Boosting Regression Trees*, etc) de acordo com métricas de interesse, como erro quadrático médio ou erro escalado.[22]

[21] Hyndman, R.J., & Athanasopoulos, G. (2021) Forecasting: Principles and Practice, 3rd Edition, OTexts: Melbourne, Australia. OTexts.com/fpp3. Acesso em 21 de fevereiro de 2022.
[22] Sobre métricas de erro, ver este link. Acesso em 18 de março de 2022.

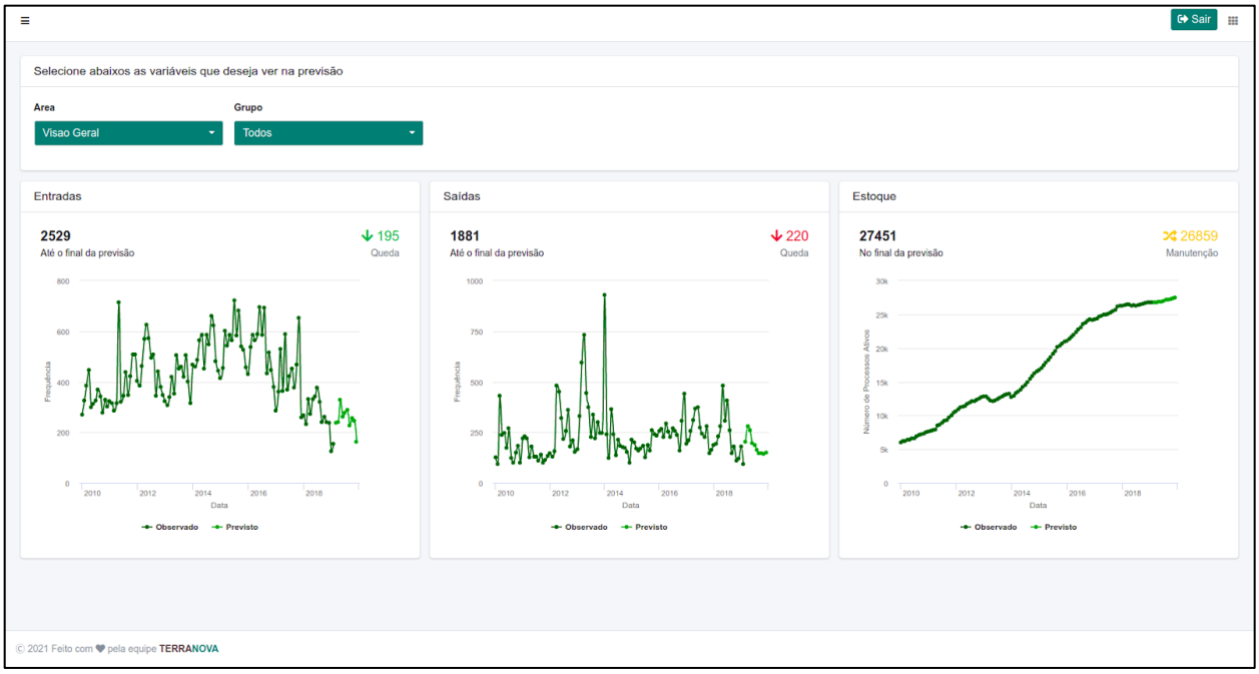

Figura 9: Estimativas de entradas e saídas de processos.

A aba "Risco" mostrará três informações principais: processos relacionados, predições e sugestões de encaminhamento em cada caso. A construção do campo de processos comparáveis do painel de jurimetria pode ser feita utilizando-se sistemas de busca modernos. A *query* (pergunta) será o texto do processo (possivelmente truncado para aumentar a eficiência da busca) e o *corpus* são todos os processos disponíveis. Para obter esses resultados, o módulo utilizará o NeuralSearchX (apresentado acima, na seção 4.1.3). Com ele, ganhamos diversas competições internacionais, inclusive a COLIEE 2021 (tarefa 2), que consiste em encontrar trechos de casos judiciais existentes que corroboram a decisão de um novo caso. Aqui, o NeuralSearchX irá fornecer uma lista de processos relacionados com base em similaridade entre petições iniciais.

A segunda parte da aba mostra as predições associadas a cada caso. A aba mostrará **as predições de um modelo de aprendizado aplicado às características de cada caso, apresentando o exame da admissibilidade e probabilidades de concessão de medida cautelar e procedência.** A prioridade do caso será definida a partir de uma ponderação entre as probabilidades de concessão de medida cautelar e procedência, com pesos a serem definidos ao longo do projeto. Os modelos serão ajustados a partir de técnicas usuais de aprendizado de

máquinas[23] incluindo, mas não se limitando, a modelos de *deep learning*. Os modelos serão ajustados a partir do histórico de processos disponível na base de dados, após devido tratamento para correção de inconsistências e *feature engineering*.[24] Os modelos terão como entrada as características dos processos (textos e metadados) e como saída um ou mais escores relacionados aos resultados de interesse.[25]

Ao final da modelagem, **apresentaremos uma matriz de erros, juntamente com métricas de impacto, para que se possa tomar a decisão do valor de corte a ser considerado em cada modelo.** Por exemplo, pode ser que o modelo com maior acurácia no geral não seja adequado pois apresenta um erro maior em processos de grande impacto. Além dos processos relacionados e dos riscos, a aba apresentará sugestões de ação em cada caso. **As sugestões serão definidas majoritariamente a partir de um fluxograma aplicando regras de negócio às predições obtidas, construído juntamente com a equipe do TCU.** As sugestões serão validadas a partir da análise de uma amostra de casos por funcionários do TCU.

Propomos que a aba de risco também apresente duas funcionalidades adicionais: 1) predição em caso hipotético e 2) interpretação de resultados. A possibilidade de obter predições em casos hipotéticos é uma ferramenta útil para estudo e entendimento do funcionamento dos modelos, já que é possível analisar como eles se comportariam em diversas situações de interesse do servidor do TCU. **Por exemplo, será possível obter uma estimativa para análise de admissibilidade e a probabilidade de concessão de medida cautelar para um caso que nunca ocorreu, mas que possui características de interesse para análise**. A Figura 10, abaixo, mostra um exemplo de predição para casos hipotéticos, desenvolvido em projetos anteriores da Terranova.

[23] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). Statistical learning. In An introduction to statistical learning (pp. 15-57). Springer, New York, NY.
[24] Brandon Butcher & Brian J. Smith (2020) Feature Engineering and Selection: A Practical Approach for Predictive Models, The American Statistician, 74:3, 308-309, DOI: 10.1080/00031305.2020.1790217.
[25] Não utilizaremos o método proposto no marco 1 pois existem muitas evidências na literatura que métodos como árvores de decisão têm desempenho superior em tarefas de predição de séries temporais (Elsayed et al, 2022).

Figura 10: Exemplo de predição de caso hipotético.

A funcionalidade de obter interpretações dos resultados permite ao servidor uma análise crítica das predições apresentadas pelo modelo. Por exemplo, é possível apresentar os *Shapley Values*[26] associados a uma predição, contendo os pesos estimados do modelo para cada variável utilizada, como o tema, os valores associados e os textos disponíveis no momento da predição. A Figura 11, abaixo, mostra um exemplo de aplicação de técnicas de interpretabilidade de modelos aplicados a um caso judicial, também já desenvolvido anteriormente pela Terranova.

[26] Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019. https://christophm.github.io/interpretable-ml-book/.

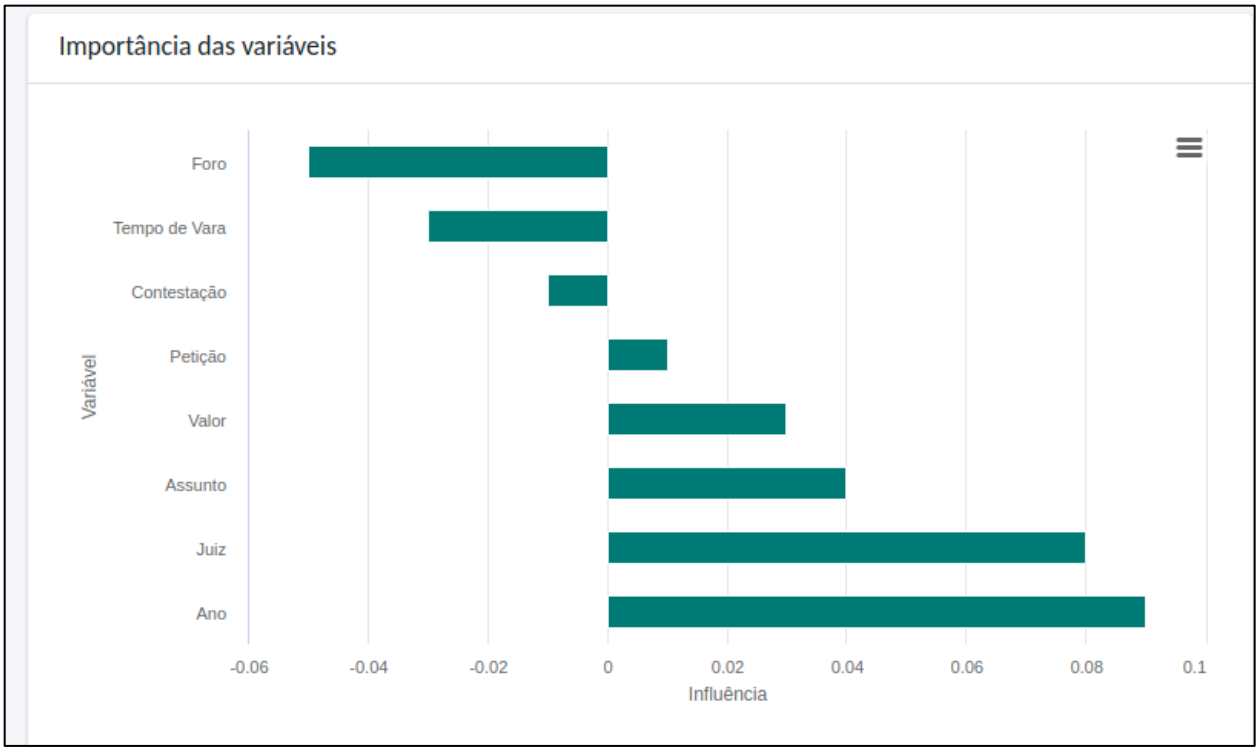

Figura 11: Exemplo de indicação de importância das variáveis para predição de encaminhamento.

4.3. Redação de Peças Processuais, Instruções e Comunicados

O terceiro marco contempla a geração automática de documentos, dentre os quais:

1. Resumo das Alegações do Autor;
2. Comunicações do Tribunal aos interessados;
3. Análise Técnica do caso;
4. Conclusão com Propostas de Encaminhamento ao relator.

Por necessitar da geração de longos textos, **este marco é o de maior dificuldade de execução.** Os modelos geradores de texto estado da arte são capazes de gerar textos fluidos e gramaticalmente corretos, porém factualmente incorretos, principalmente quando os textos são longos (Scialom et al, 2021). Mais especificamente, os modelos sofrem das chamadas "alucinações", em que produzem textos fora de contexto (Holtzman et al. 2020). A literatura atual sugere alguns métodos para mitigar o problema (ex: Wang et al 2020) porém desconhecemos trabalhos demonstrando seu sucesso em problemas reais.

Assim, usaremos a mesma metodologia descrita para o marco 1 para uma tarefa de sumarização extrativa-abstrativa. Extrativa pois a primeira subtarefa consiste em selecionar sentenças relevantes para o resumo, dado um trecho do documento como entrada. Com base nestas sentenças, o modelo gera o resumo de forma abstrativa, ou seja, produzindo palavra por palavra. Como o texto de entrada para a subtarefa abstrativa é menor e mais relevante que se usássemos o trecho original, esperamos que o modelo tenha mais facilidade de produzir resumos factualmente corretos.

4.3.1. Resumo das Alegações do Autor

Seguindo na linha da utilização de um único modelo com aprendizado *few-shot*, **também é possível elaborar o resumo da tese através de várias perguntas.** Por exemplo, o sistema irá inicialmente receber a pergunta: "*Quais são as irregularidades expostas?"*. Para cada irregularidade (IR_1,..., IR_n) dada como resposta, as seguintes perguntas seriam enviadas ao modelo: *"Quais são as proposições para a irregularidade* IR_i*?"*. Para cada proposição retornada (PR_1, ..., PR_m), a seguinte pergunta seria enviada ao modelo: *"Quais as evidências que corroboram a proposição* PR_j*?"*. Uma vez obtida as respostas, podemos estruturar o resumo no seguinte formato:

Irregularidade: IR_1
- Proposição: PR_1
  - Evidencias: $EV_1, \ldots, EV_q$

…

Irregularidade: IR_n
- Proposição: PR_m
  - Evidências: $EV_1, \ldots, EV_p$

Este processo é o mesmo do marco 1, em que o modelo recebe como entrada um prefixo contendo alguns poucos exemplos e a pergunta concatenada com um trecho da petição. Como a maioria dos trechos não tem a resposta à pergunta, o modelo deve produzir uma resposta nula (ex: "N/A") nesses casos. Para fazê-lo, o prefixo deverá ter exemplos de trechos que não contém a resposta e a respectiva saída nula desejada.

Para cada irregularidade encontrada, a concatenação de suas proposições e evidências serão usadas como entrada para o modelo gerar um resumo de forma abstrativa. **Ou seja, a tarefa é resolvida por duas subtarefas sequenciais: uma extrativa (de seleção) e outra abstrativa.** A

fase abstrativa é necessária para remover os itens que foram extraídos em duplicidade pela primeira subtarefa e para deixar a leitura mais fluída. A título ilustrativo da capacidade dos modelos *few-shot*, replicamos na Figura 12, abaixo, o exemplo da Introdução, em que a entrada do modelo é **um trecho da petição usada como exemplo no Termo de Referência desta ETEC concatenada com a pergunta *"quais são as irregularidades apontadas no trecho acima?"***.

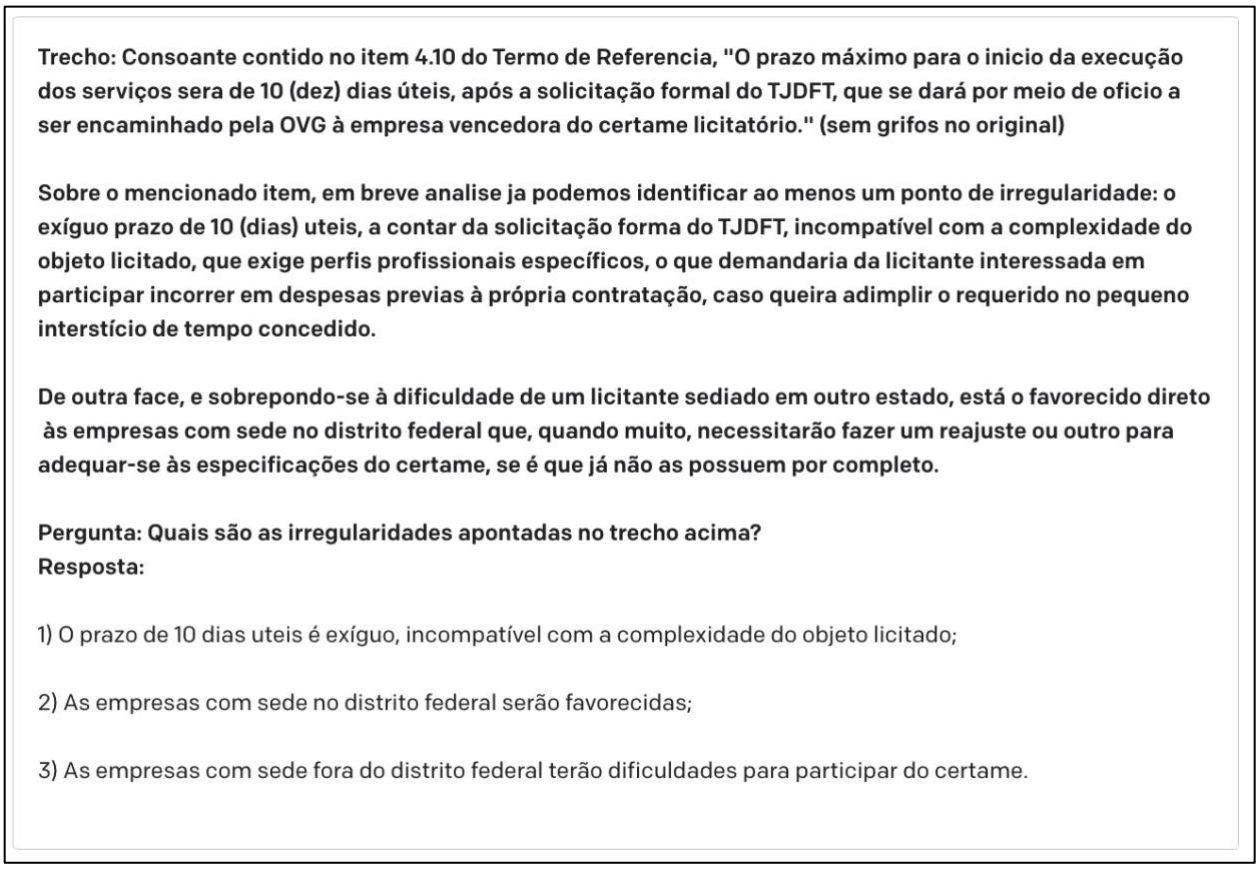

Figura 12: Exemplo de uso de modelo *zero-shot* para extrair *"Quais São as Irregularidades Apontadas em um Trecho?"*

A saída gerada pelo modelo é o texto que não está em negrito. Neste exemplo, o modelo foi capaz de identificar a irregularidade a partir de um trecho extraído do documento original. Não utilizamos nenhum exemplo de *<entrada, saída desejada>* como prefixo no *prompt*, ou seja, foi um experimento *zero-shot*. Com exemplos no *prompt*, a saída pode ser gerada em um padrão que dispensa pós-processamento.

4.3.2. Comunicação do Tribunal aos Interessados

A tarefa de escrever comunicações do Tribunal aos interessados é dividida em três subtarefas, ilustradas na figura abaixo.

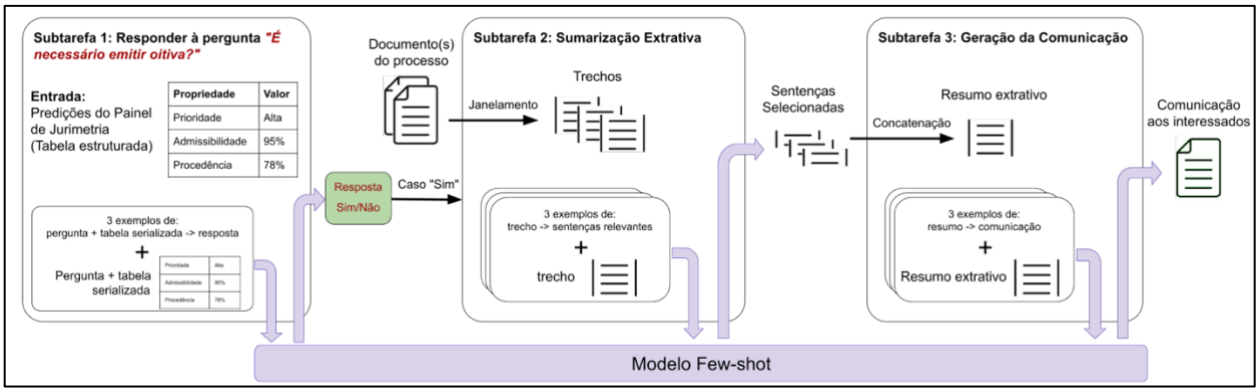

Figura 13: Ilustração do pipeline da tarefa de redação de comunicação aos interessados.

A primeira subtarefa consiste em decidir se deve ser redigida uma comunicação aos interessados a partir das predições feitas pelo módulo do painel de jurimetria. Vemos esta subtarefa como um problema de classificação binária, onde o modelo *zero-shot* tem como entrada a pergunta "*É necessário emitir uma oitiva?*" concatenada com as predições do módulo do painel de jurimetria, e a saída é uma resposta "Sim" ou "Não". As predições do painel de jurimetria estão em um formato estruturado (ex: propriedade-valor) e serão convertidas para linguagem natural usando um *template* simples. Por exemplo, se as predições para um processo foram:

Propriedade	Valor
Prioridade	Alta
Admissibilidade	95%
Procedência	78%

Os dados acima serão serializados para: *"A prioridade do processo é alta, a admissibilidade é 95% e a procedência é 78%"*. Esta conversão se faz necessária pois os modelos

de linguagem com aprendizado *few-shot* foram pré-treinados em textos em linguagem natural e podem ter uma queda de qualidade nas predições ao lidar com informação estruturada.[27]

Caso o sistema responda "Não" à pergunta, o pipeline é interrompido e uma próxima pergunta é enviada (ex: "*É necessário requerer uma inspeção?*"). Caso o sistema responda "Sim", a segunda subtarefa será disparada, na qual utilizaremos o mesmo método de geração de resumo das alegações do autor. Nesta subtarefa, primeiramente o sistema extrai sentenças relevantes à geração da comunicação (como sentenças que contém o nome e endereço das partes).

Por fim, **a última subtarefa consiste em usar as sentenças extraídas pela subtarefa anterior para redigir a comunicação. Aqui, o texto final é gerado palavra por palavra a partir das informações contidas nas sentenças extraídas.** O modelo *few-shot* é informado sobre o formato do texto de comunicação que deve ser gerado através de alguns poucos exemplos fornecidos no prompt. **Não iremos utilizar a técnica mais clássica de preenchimento de *templates* pois nossa experiência é que é difícil trabalhar com um pequeno conjunto de *templates* que satisfaça a maioria dos casos.** Na prática, o número de *templates* necessários é grande e consequentemente a lógica para decisão de qual *template* usar fica complexa, o que aumenta as chances de erros na geração do texto. O método proposto dispensa esse esforço de criação de *templates*: apenas exemplos de comunicações emitidas no passado são utilizados para criar texto dentro do formato esperado.

4.3.3. Predição de Análise Técnica e Propostas de Encaminhamento

Até agora, propusemos utilizar poucos exemplos como entrada para os modelos *few-shot* para facilitar o desenvolvimento dos vários *pipelines* que comporão este projeto. Entretanto, **devido à complexidade das tarefas de predição da análise técnica e propostas de encaminhamento, utilizaremos uma base de referência contendo centenas de exemplos de análise técnicas e propostas de encaminhamento redigidas no passado pelo TCU**. Como não é possível usar como entrada para os modelos *few-shot* centenas desses exemplos, utilizaremos os

[27] A depender do interesse do TCU, outra solução é abstrair a classificação binária da primeira tarefa e permitir que a INA[2] seja capaz de redigir qualquer documento desde que solicitado por um funcionário do TCU. Assim, a decisão de redação fica a cargo do funcionário, que a pode tomar com base nas indicações do Painel de Jurimetria e elementos externos ao sistema; nesta alternativa, portanto, a geração do documento seria adicionada via funcionalidade *point-and-click* no Painel de Jurimetria.

prompts dinâmicos descritos anteriormente, aos quais alguns poucos exemplos da base de referência, semelhantes ao texto de entrada que desejamos realizar uma predição, serão selecionados como prefixo. Para selecionar os exemplos a partir da base de referência, utilizaremos um sistema de busca moderno, como o NeuralSearchX. **Uma vez selecionados os exemplos do prompt, a INA[2] terá condições de gerar as predições da análise técnica e propostas de encaminhamento embasando-se em exemplos de textos produzidos no passado.**

Avaliação: A avaliação da qualidade dos textos gerados será feita utilizando-se métricas tradicionais de avaliação de modelos generativos como ROUGE e BLEU. É sabido, entretanto, que essas métricas têm baixa correlação com julgamentos humanos (principalmente o ROUGE). **Similarmente à estratégia para a marco 1, nossos esforços neste marco se concentrarão em realizar avaliações humanas para medir a qualidade dos textos produzidos pelo sistema** (com testes A/B, por exemplo).

4.4. Vantagens

Reconhecemos que a metodologia proposta é arrojada. Para mitigar seus riscos, executaremos um cronograma com entregas curtas e incrementais. Por exemplo, ao final do terceiro trimestre do projeto, **teremos um protótipo de extração parcial do formulário de admissibilidade usando modelos *few-shot* com os quais já temos experiência**.

Também reconhecemos que modelos *few-shot* ainda não têm o mesmo desempenho que modelos treinados com centenas ou milhares de exemplos em tarefas específicas.[28] **Entretanto, para um projeto complexo como este, entendemos que a melhor alternativa é utilizar modelos *few-shot,* já que o número de subtarefas a serem executadas por pipeline será grande.** Por exemplo, se fossemos usar modelos com treinamento específico para responder à pergunta *"A petição inicial encontra-se acompanhada de indício concernente à irregularidade ou ilegalidade apontada pelo autor?"* do exame de admissibilidade, teríamos que treinar e avaliar pelo menos quatro (4) modelos:

1. Extrair segmentos da petição inicial que mencionem a irregularidade ou ilegalidade apresentada pelo autor;

[28] Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., ... & Rush, A. M. (2021). Multitask prompted training enables zero-shot task generalization. https://arxiv.org/abs/2110.08207.

2. Classificar quais documentos do anexo podem ser usados como indícios de irregularidade;
3. Para cada documento, encontrar trechos que possam ser usados como indícios;
4. Comparar os trechos extraídos da etapa 1 e etapa 3, par-a-par, e agregar as pontuações par-a-par em uma única pontuação que representa a qualidade das evidências fornecidas.

Em artigo recente, mostramos que a utilização de modelos *sequence-to-sequence* (i.e., "entra texto, sai texto"), como a INA[2], simplifica o desenvolvimento de soluções customizadas quando comparado a *pipelines* clássicos. **Os modelos *sequence-to-sequence* têm capacidade de produzir textos prontos para serem consumidos pelo modelo da próxima etapa do pipeline, e portanto, a grande maioria das rotinas de pré e pós processamento podem ser eliminadas**. *Pipelines* clássicos, entretanto, requerem inúmeras etapas de pré e pós processamento especificamente construídas para formatar as entradas e saídas dos modelos extratores.

Anonimização dos dados: uma outra vantagem de se usar modelos *few-shot* é que fica mais fácil anonimizar os dados de treinamento, já que são poucos. É importante notar que não é necessário anonimizar os dados de validação, pois eles não são utilizados no treinamento e não fazem parte do modelo pré-treinado. Para o pré-treino não-supervisionado a ser realizado nos documentos do Tribunal, iremos anonimizar os dados da seguinte forma: primeiramente um sistema de reconhecimento de entidades nomeadas irá detectar todos os nomes de pessoas naturais em um determinado documento. Esses nomes serão mapeados para nomes fictícios, que podem ser criados através de concatenação aleatória de primeiros nomes e sobrenomes reais. Neste mapeamento, é importante substituir um nome para o seu correspondente fictício de forma constante ao longo do documento. Por exemplo, todas as menções à "João Silva" no documento original devem ser mapeadas para um mesmo nome fictício (ex: "Pedro Alves"). Isto é necessário para que o modelo de linguagem possa aprender que pessoas são referenciadas múltiplas vezes ao longo do documento. Esse mesmo processamento será, então, repetido para todos os outros dados que requerem anonimização, como CPFs e endereços.

5. Atividades

5.1. Etapa Preliminar: Reconhecimento Ótico de Caracteres (OCR)

Nesta etapa, nós construiremos um pipeline de *Optical Character Recognition* (OCR) capaz de processar qualquer formato de arquivo (.pdf, .docx, .html, .rtf, .jpeg, .odc, etc.) e retornar o texto simples (.txt) linearizado para consumo pela INA[2] e pelo Painel de Jurimetria. **As equipes da NeuralMind e da Terranova já implementaram solução semelhante** para produtos internos ou consultorias (NeuralMind NDocs® e consultorias da Terranova para Banco Bocom BBM e Recovery) e externos (Querido Diário Open Knowledge Brasil (OKBR)). Para a maioria dos casos, utilizaremos tecnologias abertas (como o Apache Tika e o Tesseract); para os documentos mais complexos, utilizaremos plataformas proprietárias (Google Cloud Vision ou Amazon Textract).[29]

5.2. Etapa: Detecção e Extração de Informações de Peças Processuais

As atividades do marco 1 serão divididas em três fases, de acordo com a complexidade de cada campo do formulário do exame de admissibilidade. **A primeira fase consistirá em extrair os campos de menor complexidade**, como responder à pergunta *"É registro de preço?"* ou *"O contrato foi assinado?"*. **A última fase será composta por campos de maior complexidade**, como responder à pergunta *"Existe algum dano irreversível para a administração pública caso o TCU não suspenda imediatamente o objeto?"*. A motivação para esta divisão é para produzirmos um produto mínimo viável (MVP) rapidamente, no início dos trabalhos, e que evoluirá progressivamente ao longo do projeto.

Cada uma das fases será composta das seguintes atividades:

1. **Divisão da tarefa em subtarefas:** Cada campo do formulário a ser extraído é visto como uma "tarefa", a qual será dividida em subtarefas de menor complexidade conforme explicado na seção de metodologia. Esta atividade terá a participação de cientistas de dados familiares com modelos *few-shot* e especialistas no domínio jurídico.
2. **Criação de exemplos *few-shot* e *datasets* de validação:** cada subtarefa requer a criação de alguns poucos exemplos (ex: 3) de pares de entrada e saída desejada que

[29] As ferramentas elencadas aqui e nas outras subseções não são taxativas; servem, apenas, de referência para que a equipe técnica do TCU avalie a adequação dos *frameworks* e sugira ajustes, se necessário.

serão prefixados ao *prompt* ingerido pelo modelo. Devido à necessidade de conhecimentos específicos da tarefa, esta atividade será realizada primariamente por especialistas do domínio jurídico. Esta atividade também envolve a criação de datasets de avaliação para cada subtarefa. Necessitaremos de pelo menos 50 exemplos de validação para cada uma delas.

3. **Codificação do *pipeline low-code*:** as saídas de uma subtarefa são usadas como entradas para outras subtarefas. Para realizar este encadeamento, iremos utilizar um *pipeline low-code* que requer apenas algumas funções em Python para formatar as entradas e saídas de cada subtarefa. Essa atividade é relativamente simples pois os modelos trabalham com textos sem formato rígido. Entretanto, ainda requer programadores com conhecimento de processamento de linguagem natural para desempenhá-la.
4. **Avaliação do *pipeline*:** Nesta atividade avaliaremos cada subtarefa do pipeline usando o *dataset* de avaliação criado na atividade 2. Também avaliaremos o pipeline fim-a-fim, ou seja, a proporção de campos dos formulários que foram preenchidos corretamente.

As ferramentas utilizadas nesta fase serão frameworks *open-source* de NLP (como o PyTorch e o BERTimbau) e ferramentas proprietárias (OpenAI GPT-3 API e HuggingFace T0). Para o encanamento e *pipeline* de dados de atividades, propomos infraestrutura distribuída de dados (Apache Spark) em serviço Google Cloud Platform (GCP), AWS ou Azure e API própria para ingestão, processamento e transferência de dados entre tarefas.

5.3. Etapa: Painel de Jurimetria

As atividades do marco 2 serão divididas em duas fases. **A primeira será dedicada ao Panorama Geral, contendo análises do histórico de casos, e a segunda, à Aba de Risco, com a aplicação dos métodos de aprendizado de máquina, priorização de processos e identificação de casos similares.**

A primeira fase, do Panorama Geral, será composta das seguintes atividades:

1. **Desenho do Panorama Geral**: Começaremos com uma proposta inicial de painel com um conjunto de visualizações e *Key Performance Indicators* (KPIs), construídos com base nas informações disponíveis. A proposta inicial será avaliada pela contratante, obtendo-se, assim, um *layout* final.
2. **Construção de um painel de inconsistências**: O painel de inconsistências é um passo intermediário para a elaboração das visualizações. É comum que as bases de dados apresentem problemas como lacunas e/ou erros lógicos entre datas, por exemplo. Tais problemas serão mapeados, corrigidos e monitorados.
3. **Construção das visualizações de dados:** As visualizações de dados definidas serão construídas, obtendo-se, assim, a versão preliminar do panorama geral. As visualizações que dependerem de modelos estatísticos (modelos de séries temporais ou análise de sobrevivência) serão acompanhadas de relatório técnico descrevendo a qualidade dos ajustes.

A segunda fase, da Aba de Risco, será composta das seguintes atividades:

1. **Desenho da Aba de Risco:** A partir da ideia inicial já definida no edital de pesquisa, faremos uma proposta inicial do *layout* com modelos *dummy*, ainda sem otimização. A proposta inicial será avaliada pela contratante, obtendo-se, assim, um *layout* final.
2. **Desenvolvimento dos modelos:** Nesta etapa, construiremos os *pipelines* de ajuste dos modelos, obtendo estimativas da qualidade do ajuste de acordo com métricas definidas em comum acordo com a contratante. Por exemplo, pode ser que a acurácia não seja a melhor métrica a ser otimizada na predição de medida cautelar, já que um falso positivo poderá acarretar maior custo que um falso negativo. No final, teremos um relatório com os principais resultados de todos os modelos em uma base de teste extraída aleatoriamente da amostra completa de casos.
3. **Inclusão dos processos relacionados:** A inclusão de processos relacionados a cada caso utilizará uma ferramenta de busca moderna, e será construída a partir da consulta a uma API que acessa os resultados da busca e alimenta a tabela de casos relacionados.
4. **Ajuste fino das classificações de priorização e sugestão de encaminhamentos**: Na última etapa da aba de risco, construiremos heurísticas para classificação da prioridade

(e.g. alta, média e baixa) de acordo com as predições dos modelos acerca do exame de admissibilidade, medida cautelar e resultado. Os encaminhamentos possíveis também serão construídos a partir de regras lógicas ou um modelo de regressão (o que funcionar melhor) utilizando as predições dos modelos como insumos. Os encaminhamentos serão validados a partir de uma amostra aleatória de casos.

Para a construção do Painel de Jurimetria, as ferramentas utilizadas serão o R Shiny e *frameworks* JavaScript (Node.js, React.js, D3.js, Highcharts), com as quais a Terranova tem ampla tem experiência para construir tanto o *back-end* quanto o *front-end* de plataformas web de visualização.

5.4. Etapa: Redação de Peças Processuais, Instruções e Comunicados

As atividades do marco 3 (Redação de Peças) são similares às do marco 1, mas que agora envolvem a geração de textos mais longos. Estas atividades também serão divididas em três fases, de acordo com a complexidade de texto a ser gerado. A primeira fase consistirá em tarefas que requerem textos gerados de menor complexidade, como gerar o resumo das alegações do autor. A última fase será composta por tarefas de maior complexidade, como a de redigir propostas de encaminhamento.

Cada uma das fases será composta das seguintes atividades:

1. **Divisão da tarefa em subtarefas:** Cada tipo de texto a ser gerado é visto como uma "tarefa", a qual será dividida em subtarefas de menor complexidade conforme explicado na seção de metodologia. Esta atividade terá a participação de cientistas de dados familiares com modelos *few-shot* e especialistas no domínio jurídico.
2. **Criação dos exemplos do prompt:** cada subtarefa requer a criação de alguns poucos exemplos (ex: 3) de pares de entrada e saída desejada que serão prefixados aos *prompts* da INA[2]. Devido à necessidade de conhecimentos específicos da tarefa, esta atividade será realizada primariamente por especialistas do domínio jurídico. Esta atividade também envolve a criação de *datasets* de avaliação para cada subtarefa. Esperamos que necessitaremos de pelo menos 50 exemplos de validação para cada subtarefa.

3. **Codificação do *pipeline low-code*:** as saídas de uma subtarefa são usadas como entradas para outras subtarefas. Essa atividade é relativamente simples pois os modelos trabalham com textos sem formato rígido. Entretanto, ainda requer programadores com conhecimento de processamento de linguagem natural para desempenhá-la.
4. **Avaliação do *pipeline*:** Nesta atividade avaliaremos cada subtarefa do pipeline usando o *dataset* de avaliação criado na atividade 2. Também avaliaremos o *pipeline* fim-a-fim: um avaliador humano dará uma nota quanto à qualidade dos textos gerados, principalmente quanto à acurácia do texto gerado (precisão) e a proporção dos fatos relevantes dos documentos de entrada que estão presentes no texto gerado (revocação).

As ferramentas desta etapa são as mesmas da etapa 1.

5.5. Etapa: Pré-treinamento de Modelos no Domínio Jurídico

Em paralelo às atividades descritas acima, realizaremos o pré-treinamento do modelo único com capacidade de aprendizado *few-shot* em textos jurídicos em português, preferencialmente usando documentos fornecidos pelo TCU. Esta atividade será executada em sua maioria por cientistas de dados com a eventual consultoria de especialistas no domínio jurídico para auxiliar na seleção de documentos com temas diversos. As ferramentas desta etapa são as mesmas da etapa 5.1 e 5.4.

6. Produtos[30]

6.1. INA^2: Modelo Unificado Generativo de Aprendizado Profundo

O principal produto desta proposta é a criação do modelo unificado, generativo, de extração de informações e redação de peças processuais - a INA^2. Concretamente, este modelo é um conjunto de arquivos com os pesos e hiperparâmetros para processar informações textuais e uma API para acesso para as diferentes tarefas e subtarefas desta encomenda tecnológica. O *MVP* da INA^2 será entregue após três trimestres da data de contratação do consórcio e será continuamente melhorada ao longo do projeto.

[30] A depender da preferência da equipe técnica do TCU, estes produtos poderão ser todos "conteinerizados" para facilitar testes e implementação em paralelo e produção ao final do projeto.

6.2. Painel de Jurimetria

O segundo produto é o Painel de Jurimetria, que reunirá as informações não-estruturadas extraídas das peças processuais, os dados (e metadados) processuais e análises descritivas e preditivas que informarão o processo decisório dos funcionários do TCU. Preliminarmente, o Painel conterá indicações de volume processual, trâmite, indicação de prioridade, admissibilidade, probabilidade de concessão de medida cautelar e procedência, matriz de risco, entre outros. Concretamente, o Painel de Jurimetria é uma série de arquivos que implementam a interface do usuário (*front-end*) e o servidor do Painel (*back-end*). Estes arquivos já conterão as funções e encanamentos necessários para consumir dados da INA^2.

6.3. Infraestrutura de Ingestão e Uso de Dados

Subsidiando os produtos principais acima, que configuram o desenvolvimento tecnológico deste edital, entendemos que há a necessidade de construção de infraestrutura de ingestão, armazenamento, processamento e transferência de dados entre os módulos que compõem a INA^2 e o Painel de Jurimetria. Concretamente, a infraestrutura de dados são arquivos de configuração de servidores de dados (a depender da escolha do TCU em solução de armazenamento) e as respectivas APIs para ingerir, processar, armazenar e transferir dados.

6.4. Plataforma Web para Extração de Informações e para Redação de Peças Processuais

Além da entrega programática do módulo de instrução assistida, o consórcio Terranova-NeuralMind propõe a construção de interface web de extração de informações e redação de peças processuais para uso interno do TCU. Este produto tem como objetivo facilitar o acesso dos funcionários que não são das áreas de tecnologia à INA^2. Estes produtos podem ser entregues independentemente ou como parte do Painel de Jurimetria, a depender do interesse do Tribunal.

7. Cronograma

(omitido)

8. Orçamento

(omitido)

9. Qualificação das Proponentes

(omitido)

10. Qualificação das Proponentes

(omitido)