Universidade Federal de Pernambuco
Centro de Informática

Graduação em Engenharia da Computação

Pedro Henrique Almeida Girão Peixinho

ESTUDO COMPARATIVO DO DESEMPENHO DE LLMS NA CLASSIFICAÇÃO DA ORIGEM DE ERROS EM PIPELINES DE IDP

Recife
2025

Pedro Henrique Almeida Girão Peixinho

ESTUDO COMPARATIVO DO DESEMPENHO DE LLMS NA CLASSIFICAÇÃO DA ORIGEM DE ERROS EM PIPELINES DE IDP

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Engenharia da Computação.

Orientador (a): Cleber Zanchettin

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Peixinho, Pedro Henrique Almeida Girão.
Estudo comparativo do desempenho de LLMs na classificação da origem de erros em pipelines de IDP / Pedro Henrique Almeida Girão Peixinho. - Recife, 2025.
22 p. : il., tab.

Orientador(a): Cleber Zanchettin
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.
Inclui referências.

1. Grandes Modelos de Linguagem. 2. Processamento Inteligente de Documentos. 3. Engenharia de prompt. 4. Diagnóstico de erros. I. Zanchettin, Cleber. (Orientação). II. Título.

000 CDD (22.ed.)

Pedro Henrique Almeida Girão Peixinho

ESTUDO COMPARATIVO DO DESEMPENHO DE LLMS NA CLASSIFICAÇÃO DA ORIGEM DE ERROS EM PIPELINES DE IDP

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Engenharia da Computação.

Aprovado em: 04/08/2025

BANCA EXAMINADORA

_____

Prof. Dr. Cleber Zanchettin (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Flavio Arthur Oliveira Santos (Examinador Interno)

Universidade Federal de Pernambuco

Estudo Comparativo do Desempenho de LLMs na Classificação da Origem de Erros em Pipelines de IDP

Pedro H. A. G. Peixinho[1], **Cleber Zanchettin**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brazil

{phagp,cz}@cin.ufpe.br

Resumo. *O diagnóstico de erros em sistemas de Processamento Inteligente de Documentos (IDP) é um processo majoritariamente manual, lento e custoso, representando um gargalo para a manutenção e evolução desses sistemas. Este trabalho investiga a automação dessa tarefa através da aplicação de Grandes Modelos de Linguagem (LLMs). Foi conduzido um estudo comparativo entre os modelos GPT-4o, Claude Sonnet 4 e Gemini 2.5 Pro para classificar a origem de falhas em três categorias — Erro de Extração, Erro de Relação e Erro da Aplicação — utilizando uma engenharia de prompts baseada em raciocínio algorítmico. Os resultados demonstram a alta eficácia da abordagem, com o modelo Gemini 2.5 Pro alcançando um F1-Score de 0.94, superando significativamente não apenas os outros LLMs, mas também um baseline heurístico (F1-Score de 0.51). A solução proposta atingiu uma redução de aproximadamente 82% no tempo de análise em comparação com um baseline temporal, validando a metodologia como uma alternativa de alto desempenho e eficiente para otimizar os ciclos de manutenção de sistemas de IDP.*

Abstract. *Error diagnosis in Intelligent Document Processing (IDP) systems is a predominantly manual, slow, and costly process, representing a bottleneck for the maintenance and evolution of these systems. This work investigates the automation of this task through the application of Large Language Models (LLMs). A comparative study was conducted between the GPT-4o, Claude Sonnet 4, and Gemini 2.5 Pro models to classify the origin of failures into three categories — Extraction Error, Relation Error, and Application Error — using a prompt engineering methodology based on algorithmic reasoning. The results demonstrate the high effectiveness of the approach, with the Gemini 2.5 Pro model achieving an F1-Score of 0.94, significantly outperforming not only the other LLMs but also a heuristic baseline (F1-Score of 0.51). The proposed solution achieved a reduction of approximately 82% in analysis time compared to a temporal baseline, validating the methodology as a high-performance and efficient alternative for optimizing the maintenance cycles of IDP systems.*

1. Introdução

O Processamento Inteligente de Documentos (IDP) é uma área da Inteligência Artificial focada na extração automática de informações de documentos não estruturados, como notas fiscais, contratos e relatórios financeiros. Para isso, um conjunto de técnicas é aplicado de forma sequencial. Primeiramente, o Reconhecimento Óptico de Caracteres (OCR) é responsável por converter a imagem do documento em texto digital. Em seguida,

com o texto já legível pela máquina, aplicam-se técnicas de Processamento de Linguagem Natural (NLP). O Reconhecimento de Entidades Nomeadas (NER) é a etapa encarregada de identificar e classificar os dados de interesse, como, por exemplo, localizar a *string* "R$ 250,00" e rotulá-la como uma entidade do tipo "valor_total". Por fim, a Extração de Relações (RE) estabelece as conexões lógicas entre essas entidades para contextualizá-las; no mesmo exemplo, essa etapa associaria a entidade "valor_total" à "data_de_vencimento" correta, distinguindo-a de outras datas no documento.

Apesar dos avanços na área, os sistemas de IDP enfrentam um desafio fundamental: a ocorrência de erros durante as etapas do processamento. Falhas podem ser introduzidas por imagens de baixa resolução, *layouts* documentais não padronizados ou pela própria ambiguidade da linguagem. A presença desses erros impõe a necessidade de um ciclo robusto de manutenção, que depende da identificação precisa de sua origem. Contudo, o diagnóstico para determinar em qual etapa uma falha ocorreu ainda é predominantemente realizado por meio de uma análise manual, um processo lento e custoso, tornando-se um grande gargalo operacional para a melhoria contínua do sistema.

Nesse contexto, os Grandes Modelos de Linguagem (LLMs) surgem como uma abordagem promissora para automatizar essa tarefa de diagnóstico. A hipótese é que a capacidade avançada desses modelos para compreensão semântica e raciocínio contextual permitiria analisar as saídas de um sistema de IDP e inferir a natureza da falha com alta confiabilidade.

Este trabalho propõe avaliar a eficácia dos LLMs na classificação da origem dos erros em um *pipeline* de IDP. Para isso, será conduzido um estudo comparativo envolvendo os modelos GPT-4o, Gemini 2.5 Pro e Claude Sonnet 4. A análise se concentrará na capacidade dos modelos de categorizar as falhas em "Erro de Extração"e "Erro de Relação", além de uma etapa posterior ao processamento do documento: a etapa de aplicação de regras de negócio ("Erro da Aplicação"). Essa análise fornecerá um panorama claro sobre a viabilidade e a eficiência desta metodologia para otimizar a melhoria contínua de sistemas de IDP.

2. Trabalhos Relacionados

Esta seção revisa os conceitos que fundamentam este trabalho, abordando o Processamento Inteligente de Documentos (IDP), os Grandes Modelos de Linguagem (LLMs) e as técnicas atuais para diagnóstico de erros utilizando LLMs.

2.1. Processamento Inteligente de Documentos

As abordagens modernas de IDP superaram o OCR tradicional ao empregar modelos multimodais que compreendem a estrutura visual do documento. Arquiteturas como LayoutLM [1] e suas variantes se tornaram um padrão ao processar simultaneamente informações textuais e de *layout*. Paralelamente, o modelo LAMBERT [2] otimiza a extração em *layouts* complexos por meio de uma técnica de rasterização, na qual as coordenadas 2D de cada palavra, obtidas por meio de um OCR, são usadas para gerar um mapa de *features* que captura a estrutura espacial do documento. Apesar dos avanços, a complexidade e variedade de *layouts* ainda geram erros, mantendo a validação manual como um gargalo operacional que justifica a busca por métodos de diagnóstico mais eficientes.

2.2. Grandes Modelos de Linguagem

A ascensão dos LLMs foi possibilitada pela arquitetura Transformer [3], que impulsionou o desenvolvimento desses modelos em grande escala. Essa evolução culminou em modelos com a capacidade de aprendizado em contexto (*in-context learning*), popularizada pelo GPT-3 [4]. Essa habilidade de executar novas tarefas a partir de instruções e poucos exemplos (*zero-shot e few-shot learning*), sem a necessidade de um *fine-tuning* específico, é a base da metodologia que será utilizada para o diagnóstico de erros.

2.3. Diagnóstico de Erros com LLMs

O uso de LLMs para tarefas de avaliação é uma área emergente, exemplificada pela abordagem "LLM-as-a-Judge"[5], na qual um modelo avalia a saída de outro. No entanto, a aplicação dessas técnicas para um diagnóstico detalhado que identifique a etapa de origem de uma falha dentro de um *pipeline* de IDP é uma área ainda pouco explorada e que este trabalho se propõe a investigar.

3. Metodologia

3.1. Base de dados

Nesta seção, serão detalhadas a origem, a estrutura e as características do conjunto de dados que servirá de base para o desenvolvimento e validação das avaliações realizadas neste trabalho.

3.1.1. Origem dos dados

O conjunto de dados foi obtido a partir de um sistema de Processamento Inteligente de Documentos (IDP) em ambiente de produção, que processa documentos de Balanço Patrimonial (BP) e Demonstração do Resultado do Exercício (DRE). O fluxo de geração de dados ocorre da seguinte forma:

1. **Submissão:** Um usuário submete um documento ao sistema (balanço, DRE ou um documento que contenha ambos simultaneamente).
2. **Extração Automatizada:** A plataforma de IDP realiza a extração automática das informações contidas no documento, aplicando modelos de OCR, NER, RE e regras de negócio pré-definidas.
3. **Validação Humana:** O resultado extraído pelo sistema é apresentado a um usuário para validação. O usuário tem a prerrogativa de corrigir qualquer campo que tenha sido extraído de forma incorreta. O sistema armazena tanto o valor originalmente extraído pela automação quanto o valor final validado (ou corrigido) pelo usuário.

Esse processo de validação humana garante a criação de um *Ground Truth* (gabarito) para cada campo, permitindo a rotulagem dos dados gerados como "corretos"ou "incorretos".

3.1.2. Estrutura e Atributos

Os conjuntos de dados de origem consistem em três relatórios, em formato Excel, em que cada linha de cada relatório corresponde a um campo extraído de um único documento.

Cada registro contém os metadados do documento, os valores extraídos pela automação e os valores validados manualmente.

Os dados foram coletados durante três períodos distintos de operação do sistema, totalizando 62.816 amostras. A distribuição da coleta ocorreu da seguinte forma:

1. 28/06/2024 e 05/07/2024, contendo 2392 amostras.
2. 14/08/2024 e 22/08/2024, contendo 572 amostras.
3. 01/04/2025 e 30/06/2025, contendo 59852 amostras.

A Tabela 1 detalha os atributos do sistema, utilizando a nomenclatura original das colunas dos arquivos-fonte para garantir a rastreabilidade.

Tabela 1. Descrição dos atributos da base de dados.

Atributo	Descrição
Nome do documento	Nome do documento processado.
Ano da extração	Ano ao qual o valor se refere dentro do documento.
Código do documento	Código único do documento processado.
Nome do campo	Nome do campo sendo observado.
Valor extraído	Valor extraído pelo sistema.
Valor esperado	Valor validado ou corrigido pelo usuário (*ground truth*).

Adicionalmente, os campos podem ser classificados de acordo com sua forma de composição:

1. **Valor direto:** Valor de uma *label* extraída.
2. **Maior valor:** Maior valor entre um conjunto de *labels* extraídas.
3. **Soma:** Soma dos valores de um conjunto de *labels* extraídas.

Essa distinção é relevante, pois o tipo do erro pode estar associado à forma como o campo é composto.

3.1.3. Preparação dos dados

O primeiro passo do pré-processamento consistiu em filtrar o *dataset* completo para isolar apenas os registros em que o valor extraído divergia do valor esperado, indicando um erro do sistema. A partir disso, uma amostra de 185 registros foi extraída e manualmente classificada de acordo com o tipo de erro encontrado dentre três tipologias predefinidas:

1. **Erro de Extração:** Uma ou mais *labels* que compõem o valor esperado do campo não foram extraídas ou foram classificadas incorretamente. (Ex: O valor de um ”ativo”de um balanço patrimonial foi classificado como um ”passivo”).
2. **Erro de Relação:** Uma ou mais *labels* que compõem o valor esperado do campo foram relacionadas com a data (ano ou período) incorreta. (Ex: O valor de um ”ativo”de 2023 de um balanço patrimonial foi relacionado com o ano de 2024).
3. **Erro da Aplicação:** Os valores foram extraídos corretamente, mas a regra de negócio está mapeada erroneamente. (Ex: O campo ”despesas_operacionais”de um balanço patrimonial deveria ser formado pelo maior valor entre ”despesas”e ”despesas_gerais_e_administrativas”, mas o sistema retornou o menor valor entre eles).

Posteriormente, os registros classificados em 2024 foram revisados, e erros de classificação manual foram corrigidos. Durante esta etapa, também foram identificados registros cujo *ground truth* estava incorreto; estes foram removidos para não comprometer a avaliação dos modelos.

Por fim, para complementar a análise do desempenho dos modelos, foram adicionadas duas novas colunas: ”Composição”, que categoriza o tipo de formação do campo (definidos na Seção 3.1.2), e ”Soma”, uma variável booleana que indica se o valor esperado foi composto a partir de uma soma de *labels*.

O processo de limpeza resultou em um conjunto de dados final com 155 amostras, distribuídas em três classes: ”Erro da Aplicação”(95 amostras, 61%), ”Erro de Relação”(34 amostras, 22%) e ”Erro de Extração”(26 amostras, 17%). Posteriormente, o conjunto foi submetido a uma divisão estratificada em duas partições:

- **Conjunto de Treino (15 amostras):** Utilizado exclusivamente para o desenvolvimento e a otimização dos prompts (*prompt engineering*).
- **Conjunto de Teste (140 amostras):** Utilizado exclusivamente para a avaliação final e comparação do desempenho dos modelos selecionados.

3.2. Arquitetura do *Pipeline* do Sistema Analisado

O sistema a ser analisado emprega um *pipeline* sequencial de quatro etapas. As três primeiras compõem o núcleo de IDP, que transforma a imagem de um documento em dados estruturados, enquanto a etapa final aplica regras de negócio para gerar o resultado. As tecnologias utilizadas em cada etapa são descritas abaixo:

1. **Reconhecimento Óptico de Caracteres (OCR):** Primeiramente, a imagem do documento é processada pelo serviço Azure Cognitive Services for Vision (v3.2) [6], que converte a informação visual em uma camada de texto digital.
2. **Reconhecimento de Entidades Nomeadas (NER):** O texto digitalizado alimenta um modelo de linguagem baseado na arquitetura LAMBERT, que é responsável por identificar e classificar as entidades de interesse.
3. **Extração de Relações (RE):** Um algoritmo personalizado, desenvolvido em Python, é executado para estabelecer as relações semânticas entre as entidades identificadas, como associar um valor monetário ao seu respectivo ano.
4. **Aplicação de Regras de Negócio:** Por fim, um serviço desenvolvido em Java aplica um conjunto de regras de negócio, gerando o resultado que será apresentado ao usuário.

3.3. Modelos

Para a tarefa de classificação de erros, este estudo avaliou e comparou o desempenho de três modelos de linguagem de grande porte (Large Language Models) de fronteira, todos acessados por meio de suas respectivas APIs comerciais.

- **GPT-4o [7]:** Desenvolvido pela OpenAI, o GPT-4o é um modelo multimodal de última geração que se destaca por sua capacidade de raciocínio avançado e de seguir instruções complexas. Com uma janela de contexto de 128.000 *tokens*, ele é capaz de processar informações extensas. A escolha do GPT-4o se deve à sua performance de estado da arte em tarefas de compreensão e lógica, características

essenciais para discernir as sutilezas entre erros de extração, relação e aplicação de regras de negócio. O modelo foi acessado via API da OpenAI, utilizando a versão gpt-4o.

- **Claude Sonnet 4 [8]:** O Claude Sonnet 4 da Anthropic é um modelo projetado para oferecer um equilíbrio entre inteligência, velocidade e custo-efetividade, superando modelos mais caros em diversos *benchmarks* de raciocínio e conhecimento. Sua janela de contexto de 200.000 *tokens* e sua proficiência em analisar artefatos complexos o tornam um candidato robusto para a análise de documentos financeiros. Foi selecionado por representar uma opção pragmática e de alto desempenho. O acesso foi realizado pela API da Anthropic, com a versão claude-sonnet-4-20250514.
- **Gemini 2.5 Pro [9]:** O Gemini 2.5 Pro, do Google, é um modelo multimodal notável por sua arquitetura otimizada para o processamento de contextos extremamente longos, com capacidade para até 1 milhão de *tokens*. Embora o prompt utilizado neste trabalho seja conciso, a arquitetura do Gemini 2.5 Pro, desenvolvida para tarefas de alta recuperação de informação em grandes volumes de dados, é relevante. Foi escolhido para avaliar como um modelo especializado em ”visão de longo alcance”lida com a tarefa de identificar erros cujo contexto pode estar implícito em diferentes partes do documento. O modelo foi acessado por meio da API do Google AI Studio (utilizando o nível gratuito), na versão gemini-2.5-pro.

3.4. Engenharia de *prompt*

A engenharia de *prompts* foi um componente crucial deste trabalho, planejada para fornecer todo o contexto necessário aos modelos e, ao mesmo tempo, guiá-los para que sigam uma lógica de raciocínio pré-definida. Para isso, foram desenvolvidos e avaliados dois *prompts* distintos, que implementam as abordagens *zero-shot* e *3-shot (few-shot)*.

No desenvolvimento do *prompt*, foram empregados diversos componentes estratégicos:

- **Definição da Tarefa e Persona.** Foi atribuída ao modelo a persona de um analista de erros em um sistema de IDP, estabelecendo um contexto profissional e direcionando seu foco para uma análise técnica. A tarefa é explicitamente definida como a identificação da causa raiz de uma falha.
- **Instrução por Raciocínio Algorítmico.** Este é o componente central do *prompt*. Em vez de pedir uma classificação direta, o modelo é instruído a seguir uma lógica hierárquica e rigorosa em três passos. Essa técnica, análoga ao *Chain of Thought* [10], força o modelo a replicar um fluxo de diagnóstico humano:
    1. **Análise de Extração:** Verificar se as *labels* necessárias foram extraídas.
    2. **Análise de Relação:** Verificar se as *labels* extraídas foram associadas à data correta.
    3. **Análise da Aplicação:** Se os passos anteriores estiverem corretos, inferir que a falha está na regra de negócio final.

  Essa abordagem estruturada reduz a ambiguidade e aumenta drasticamente a confiabilidade do diagnóstico.
- **Fornecimento de Contexto do Erro.** Para que o modelo compreenda o cenário completo, uma descrição detalhada do erro é oferecida, informando o campo, a forma como ele é composto, a data e o valor obtido e esperado.

- **Apresentação de Evidências Estruturadas.** O modelo recebe as saídas processadas do sistema de IDP, que servem como evidências para a análise. As informações são estruturadas para maximizar a clareza:
  - **Saída da Extração de Entidades:** O texto do documento é fornecido com as entidades rotuladas (exemplo: <label=ativo>1.000,00), preservando o *layout* original. Isso oferece contexto visual e semântico sobre o que foi extraído.
  - **Saída da Extração de Relações:** As associações entre datas e *labels* são apresentadas em formato de dicionário ('<data>2024': ['<label=ativo>1.000,00', '<label=passivo>2.000,00', ...']). Essa estrutura simplifica a verificação das relações lógicas pelo modelo.
- **Restrição do Formato de Saída.** Para garantir consistência e facilitar a avaliação automática, o modelo é instruído a retornar somente a *string* correspondente ao tipo de erro, sem qualquer formatação ou texto adicional.

Os *prompts* foram refinados de forma iterativa por meio de testes manuais no conjunto de treino. A versão final do prompt para a abordagem *zero-shot* é detalhada na Figura 1. Para a abordagem *few-shot*, essa mesma estrutura base foi estendida com a adição de três exemplos ao final do prompt, um para cada categoria de erro. Os exemplos adicionados são ilustrados na Figura 2.

3.5. Linhas de base (*baselines*)

Foram criadas duas linhas de base para comparar com o desempenho dos modelos avaliados:

A primeira foi uma avaliação temporal, estabelecida a partir do tempo médio levado para diagnosticar cada erro manualmente. Esse *baseline* representa um ponto de referência para quantificar o ganho de eficiência da automação.

A segunda foi uma avaliação do desempenho (*baseline* heurístico). Para isso, foi desenvolvido um classificador heurístico em Python que simula a mesma lógica hierárquica de diagnóstico descrita no *prompt*, detalhada na Seção 3.4. Essa abordagem representa uma solução tradicional baseada em regras, servindo como um ponto de comparação fundamental para a performance dos LLMs.

3.6. Experimentos

Esta seção detalha as métricas de avaliação utilizadas, a configuração dos experimentos e os diferentes cenários de análise para investigar as capacidades de cada modelo.

3.6.1. Métricas de Avaliação

Dado que o conjunto de dados de teste é desbalanceado, de acordo com a distribuição de classes descrita na Seção 3.1.3, a seleção de métricas foi focada em avaliar o desempenho de forma justa, sem ser influenciada pela classe majoritária.

As seguintes métricas foram utilizadas:

- **Matriz de Confusão:** Utilizada como ferramenta primária para visualização do desempenho. Ela permite uma análise qualitativa dos erros, mostrando quais classes são mais frequentemente confundidas entre si. É a base para o cálculo das demais métricas.

Você irá analisar erros que ocorreram em um sistema de IDP (Intelligent Document Processing). Nele, são retornados valores correspondentes a campos presentes nos documentos. Esses campos podem ser compostos pela soma de um grupo de labels, pelo maior valor em um grupo de labels ou sendo o valor de somente uma label.
O sistema extrai labels, relaciona essas labels com suas datas correspondentes e forma os campos de cada data.
Vou fornecer a extração de um documento para ser analisada. O texto entre <>corresponde a uma label.
Exemplo: "<label=ativo>1.000,00"indica que o valor "1.000,00"foi indexado como "ativo".
Os relacionamentos serão feitos somente com os textos da label "data".
Quando <data>for Saldo Anterior ou Exercício Anterior, ela irá corresponder ao menor ano do documento.
Quando <data>for Saldo Atual ou Exercício Atual, ela irá corresponder ao maior ano do documento.
Também vou fornecer a saída da extração de relações, que será um dicionário onde as chaves são as datas e os valores são uma lista das labels correspondentes a essas datas.
Para determinar o tipo do erro, você deve seguir a seguinte lógica hierárquica e rigorosa:
—
Passo 0: Pré-Análise de Caso Especial (Valor Esperado = 0)
* 1. Verifique a lista 'labels de interesse que foram extraídas'. Ela contém uma label para o campo com o valor "0"?
* Se SIM (o sistema extraiu um "0"explícito): O "0"é um valor válido. Trate-o como qualquer outro valor esperado e pule diretamente para o Passo 2 (Análise de Relação) para verificar se este "0"foi relacionado corretamente.
* Se NÃO (o sistema NÃO extraiu um "0"explícito): Isso significa que o valor esperado de "0"representa a ausência de um valor. Agora, verifique a lista 'labels de interesse' novamente:
* Se a lista contém QUALQUER valor (não-zero) para as labels do campo, o erro é "Erro de Extração". (O sistema extraiu um valor quando não deveria).
* Se a lista está VAZIA para as labels do campo, e o 'Valor obtido' é diferente de 0, o erro é "Erro da Aplicação". (A extração agiu corretamente, mas a aplicação falhou).
—
Passo 1: Análise de Extração (Quando o Valor Esperado é diferente de 0)
* Verifique a lista 'labels de interesse que foram extraídas'.
* A label correspondente ao 'Valor esperado' (ou os componentes que o formam) foi extraída?
* Se a resposta for NÃO, o erro é "Erro de Extração".
* Se a resposta for SIM, prossiga para o Passo 2.

Passo 2: Análise de Relação
* Verifique a 'saída da extração de relações'.
* A label (ou os componentes) do 'Valor esperado' foi corretamente associada à 'Data/Ano' em questão?
* Se a resposta for NÃO, o erro é "Erro de Relação".
* Se a resposta for SIM, prossiga para o Passo 3.

Passo 3: Análise da Aplicação
* Se a extração e a relação do valor esperado foram corretas, mas o 'Valor obtido' ainda está diferente do 'Valor esperado', a falha ocorreu na lógica final do sistema (ex: falhou ao aplicar a soma/máximo, usou um valor de outra data/coluna, etc.).
* Neste caso, o erro é "Erro da Aplicação".
—
Como resposta, retorne SOMENTE o tipo do erro: "Erro de Extração", "Erro de Relação"ou "Erro da Aplicação". Não inclua nenhuma outra palavra, explicação ou formatação.

Figura 1. Prompt zero-shot

- **Acurácia Balanceada (*Balanced Accuracy*):** Corresponde à média aritmética do *recall* de cada classe. Foi escolhida por neutralizar o efeito do desbalanceamento, tratando a performance em classes minoritárias com o mesmo peso da classe majoritária.
- **F1-Score (Macro):** Calculado como a média harmônica entre precisão e *recall*. Utilizou-se a média macro (*macro average*), que calcula a métrica para cada classe individualmente e depois tira a média. Isso também garante que todas as classes

A seguir, vou fornecer 3 exemplos de como a análise deve ser feita.
— INÍCIO DOS EXEMPLOS —
Exemplo 1
[PROMPT DO USUÁRIO]
Descrição do erro:
Campo: contas_receber, formado pela soma das labels [”contas_a_receber->ativo_circulante”, ”clientes_contas_a_receber->ativo_circulante”, ”clientes->ativo_circulante”, ”clientes->ativo_nao_circulante”, ”duplicatas_a_receber->ativo_circulante”, ”duplicatas_descontadas->ativo_circulante”]
Ano/Data: 2024
Valor obtido: 1000.00
Valor esperado: 1500.00
Segue a saída da extração, com as informações do documento:
Ativo Circulante - <data>2024
Clientes: <label=clientes->ativo_circulante>1.000
Ativo Não Circulante
Clientes: 500
Seguem as labels de interesse que foram extraídas:
[’<label=clientes->ativo_circulante>1.000’]
Por fim, segue a saída da extração de relações:
’<data>2024’: [’<label=clientes->ativo_circulante>1.000’]
—
[RESPOSTA DO ASSISTENTE]
Erro de Extração
Exemplo 2
[PROMPT DO USUÁRIO]
Descrição do erro:
Campo: fornecedores, formado somente pela label fornecedores->passivo_circulante
Ano/Data: 2024
Valor obtido: 0.00
Valor esperado: 10000.00
Segue a saída da extração, com as informações do documento:
PASSIVO <data>2024 <data>2023
Fornecedores <label=fornecedores->passivo_circulante>10.000 <label=fornecedores->passivo_circulante>8.000
Seguem as labels de interesse que foram extraídas:
[’<label=fornecedores->passivo_circulante>10.000’, ’<label=fornecedores->passivo_circulante>8.000’]
Por fim, segue a saída da extração de relações:
’<data>2024’: [], ’<data>2023’: [’<label=fornecedores->passivo_circulante>10.000’, ’<label=fornecedores->passivo_circulante>8.000’]
—
[RESPOSTA DO ASSISTENTE]
Erro de Relação
Exemplo 3
[PROMPT DO USUÁRIO]
Descrição do erro:
Campo: endividamento_curto_prazo, formado pelo maior valor entre as labels [’emprestimos_e_financiamentos->endividamento_curto_prazo_(passivo_circulante)’, ’financiamentos->endividamento_curto_prazo_(passivo_circulante)’, ”instituicoes_financeiras->endividamento_curto_prazo_(passivo_circulante)”, ”capital_de_giro->endividamento_curto_prazo_(passivo_circulante)”]
Ano/Data: 2023
Valor obtido: 2000.00
Valor esperado: 5000.00
Segue a saída da extração, com as informações do documento:
PASSIVO - <data>2023
Empréstimos: <label=emprestimos_e_financiamentos->endividamento_curto_prazo_(passivo_circulante)>5.000
Financiamentos: <label=financiamentos->endividamento_curto_prazo_(passivo_circulante)>2.000
Seguem as labels de interesse que foram extraídas:
[’<label=emprestimos_e_financiamentos->endividamento_curto_prazo_(passivo_circulante)>5.000’,
’<label=financiamentos->endividamento_curto_prazo_(passivo_circulante)>2.000’]
Por fim, segue a saída da extração de relações:
’<data>2023’: [’<label=emprestimos_e_financiamentos->endividamento_curto_prazo_(passivo_circulante)>5.000’,
’<label=financiamentos->endividamento_curto_prazo_(passivo_circulante)>2.000’]
—
[RESPOSTA DO ASSISTENTE]
Erro da Aplicação
— FIM DOS EXEMPLOS —

Figura 2. Exemplos da abordagem 3-shot

contribuam igualmente para o resultado final, independentemente de seu tamanho.

- **Matthews Correlation Coefficient (MCC):** Selecionado por ser considerado uma das métricas mais robustas e informativas para classificação desbalanceada [11]. O MCC mede a qualidade da classificação através de um coeficiente de correlação entre os valores reais e os previstos, considerando proporcionalmente todas as classes e produzindo uma pontuação que só é alta se o modelo tiver um bom desempenho em todas elas.
- **Tempo de Execução (Latência):** Sendo a análise de eficiência uma contribuição central deste trabalho, foi aferido o tempo médio de resposta (latência) de cada modelo para processar uma única requisição. A comparação desses valores com o tempo de classificação manual (*baseline* temporal) será explorada na discussão dos resultados.

3.6.2. Configuração Experimental

Para garantir a validade e a possibilidade de reprodução dos resultados, todos os experimentos foram conduzidos sob uma configuração controlada. Foram utilizados os modelos GPT-4o, Claude Sonnet 4 e Gemini 2.5 Pro, acessados por meio de suas respectivas APIs oficiais.

A fim de assegurar resultados determinísticos e consistentes, o parâmetro de temperatura de todos os modelos foi fixado em 0.0. Esta configuração minimiza a aleatoriedade na geração da resposta, garantindo que a saída seja baseada estritamente na lógica do *prompt*. Os demais parâmetros foram mantidos em seus valores padrão fornecidos pelas APIs. Para a execução, cada um dos 140 registros do conjunto de teste foi submetido uma única vez a cada modelo, tanto para a abordagem *zero-shot* quanto para a *few-shot*.

3.6.3. Cenários de Avaliação

Para realizar uma análise aprofundada das capacidades de cada modelo, o desempenho foi avaliado sob cinco cenários distintos. O primeiro cenário oferece uma visão panorâmica e completa, enquanto os três cenários subsequentes investigam aspectos específicos do problema, utilizando o F1-Score como métrica principal para garantir clareza e concisão nas comparações. Por fim, o último cenário avalia o custo de cada modelo.

A avaliação foi aplicada a ambas as abordagens de *prompt* (*zero-shot* e *few-shot*):

1. **Análise de Desempenho Geral e Eficiência:** Para esta análise principal, empregou-se o conjunto completo de métricas para construir um cenário comparativo do desempenho de cada modelo.
2. **Desempenho por Classe de Erro:** Esta análise granular, baseada no F1-Score individual para cada classe, revela se um modelo possui maior dificuldade ou proficiência em diagnosticar um tipo específico de falha.
3. **Análise de Influência por Tipo de Composição:** Este experimento investiga como a complexidade da regra de negócio do campo afeta o diagnóstico. O conjunto de teste foi segmentado de acordo com o tipo de composição do valor esperado (valor direto, soma ou maior valor). O F1-Score foi então calculado para cada um desses subconjuntos, medindo o impacto da complexidade da tarefa no desempenho.

4. **Análise de Robustez em Cenários de Agregação:** Foi avaliado como os modelos se comportaram quando o valor esperado era composto pela soma de múltiplos valores. O F1-Score foi novamente utilizado para quantificar a diferença de performance e avaliar a robustez dos modelos nesta condição, que exige uma agregação mais complexa de informações.
5. **Análise de Custo:** Para contextualizar a viabilidade econômica de cada solução, foi realizada uma análise de custo, apresentando e comparando os preços públicos das APIs de cada modelo, cotados por milhão de tokens de entrada e saída.

4. Resultados

Esta seção apresenta e analisa os resultados obtidos a partir da execução dos experimentos. A análise está estruturada de acordo com os cenários de avaliação propostos, iniciando com uma visão geral do desempenho dos modelos e, em seguida, aprofundando em análises granulares para entender suas capacidades em contextos específicos.

4.1. Análise de Desempenho Geral e Eficiência

O primeiro cenário estabelece a performance de base comparativa, avaliando o desempenho agregado e a eficiência de cada modelo. Os resultados consolidados, abrangendo todas as métricas de avaliação, são apresentados na Tabela 2.

Tabela 2. Resultados gerais de desempenho e eficiência por modelo e abordagem. Os melhores resultados em cada métrica de acurácia estão em negrito.

Modelo	Abordagem	F1-Score	Acc. Balanceada	MCC	Latência (s)
Baseline de Desempenho	-	0.51	0.63	0.41	-
Gemini 2.5 Pro	*Zero-Shot*	0.93	**0.94**	**0.91**	20.28
	3-Shot	**0.94**	**0.94**	**0.91**	20.94
GPT-4o	*Zero-Shot*	0.52	0.60	0.34	**2.34**
	3-Shot	0.55	0.59	0.32	2.57
Claude Sonnet 4	*Zero-Shot*	0.65	0.64	0.47	2.46
	3-Shot	0.67	0.68	0.50	2.74

A análise da Tabela 2 revela uma clara superioridade do modelo Gemini 2.5 Pro, que atingiu um desempenho de estado da arte na tarefa, com um F1-Score de 0.94 e um MCC de 0.91. Esses valores não são apenas significativamente mais altos que os dos concorrentes, mas também demonstram um grande salto de performance em relação ao *baseline* de desempenho, com F1-Score de 0.51 (84% de aumento). Isso indica que, embora a lógica hierárquica seja funcional, a capacidade de compreensão contextual do LLM é crucial para lidar com as nuances dos dados, resultando em uma classificação muito mais fidedigna.

A margem de desempenho sobre os demais modelos também é expressiva: na abordagem 3-shot, o Gemini supera o Claude Sonnet 4, com F1 de 0.67 e MCC de 0.50 (40% e 82% de aumento, respectivamente), e o GPT-4o, com F1 de 0.55 e MCC de 0.32 (71% e 184% de aumento, respectivamente).

Um achado relevante é o impacto modesto da abordagem *3-shot*. A adição de três exemplos resultou em ganhos de performance marginais ou mínimos para todos os modelos, ao mesmo tempo que aumentou a latência e o custo devido ao maior número de *tokens* no *prompt*. Isso sugere que, para esta tarefa de raciocínio estruturado, a clareza da instrução *zero-shot* é o fator dominante, tornando a abordagem *few-shot* menos prática e com menor custo-benefício para uma implementação em produção.

Em termos de eficiência, enquanto os modelos da OpenAI e Anthropic apresentaram as menores latências (aprox. 2.5s), o desempenho do Gemini (aprox. 21s) é particularmente notável quando comparado ao *baseline* temporal, medido em aproximadamente 2 minutos (120 segundos), representando uma redução de 82% no tempo de análise. É importante ressaltar que a latência do Gemini pode ter sido influenciada pelo uso da API em seu nível gratuito, que pode apresentar performance diferente das versões comerciais pagas. Este resultado valida a principal contribuição do trabalho: a automação da tarefa não só é viável em termos de desempenho, mas também oferece um alto ganho de eficiência, liberando recursos humanos para atividades de maior valor agregado.

A análise visual das matrizes de confusão permite uma avaliação detalhada do desempenho de cada modelo. A matriz do Gemini (Figura 4) mostra uma concentração quase perfeita na diagonal principal, com poucos erros residuais. Em contrapartida, os outros modelos apresentam padrões de erro claros: o GPT-4o (Figura 6) demonstra uma confusão generalizada, enquanto o Claude Sonnet 4 (Figura 5) tende a confundir ”Erro de Aplicação”com ”Erro de Relação”, indicando menor confiabilidade. A matriz do *baseline* de desempenho (Figura 3) revela um viés para a predição da classe ”Erro de Extração”, que foi incorretamente atribuída a um grande volume de amostras de outras categorias.

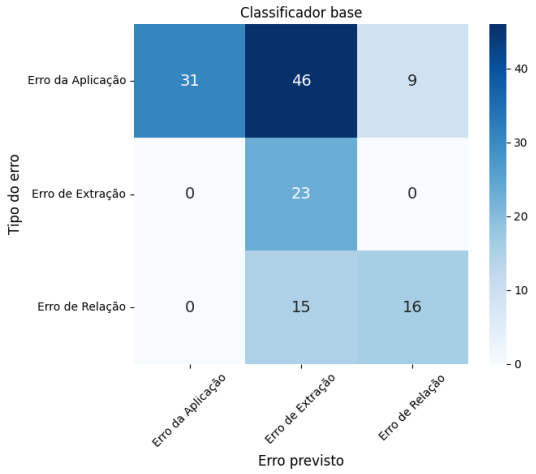

Figura 3. Matriz de confusão para a linha de base de desempenho.

4.2. Análise de Desempenho por Cenários Específicos

Para entender com mais profundidade o comportamento dos modelos, as análises a seguir segmentam os resultados utilizando o F1-Score como métrica principal, conforme planejado na metodologia.

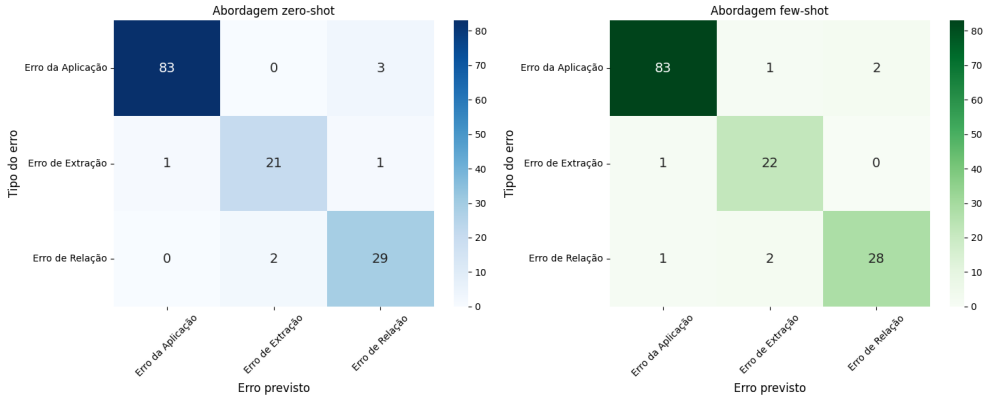

Figura 4. Matrizes de confusão para o modelo Gemini 2.5 Pro (*Zero-Shot* à esquerda, *3-Shot* à direita).

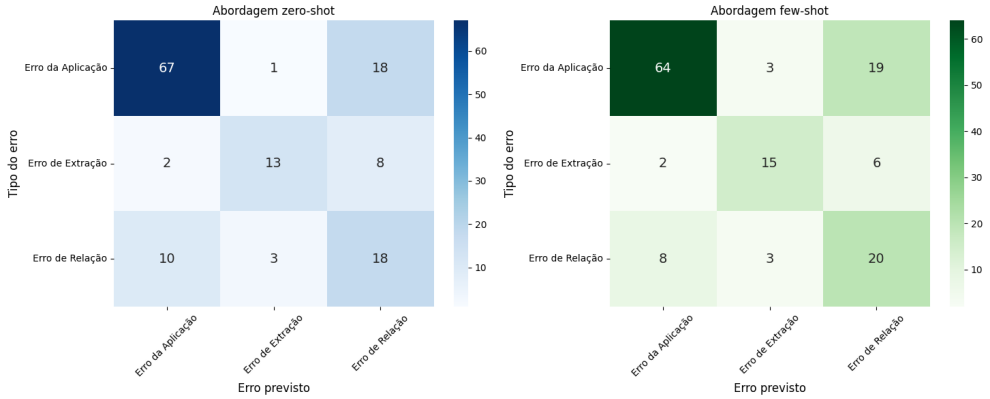

Figura 5. Matrizes de confusão para o modelo Claude Sonnet 4 (*Zero-Shot* à esquerda, *3-Shot* à direita).

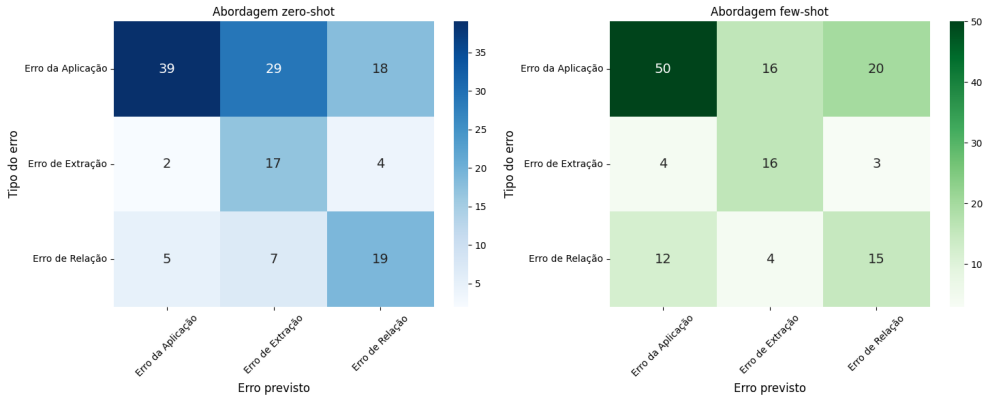

Figura 6. Matrizes de confusão para o modelo GPT-4o (*Zero-Shot* à esquerda, *3-Shot* à direita).

4.2.1. Desempenho por Classe de Erro

A Tabela 3 detalha a capacidade de cada modelo em diagnosticar cada um dos três tipos de erro, permitindo identificar especialidades ou dificuldades em cada categoria.

A análise por classe evidencia o perfil de desempenho de cada modelo. O Gemini demonstrou uma performance excepcional e equilibrada, com F1-Score acima de 0.91 em

Tabela 3. Comparativo do F1-Score por classe de erro. Os melhores resultados em cada classe estão em negrito.

Modelo	Abordagem	Erro da Aplicação	Erro de Extração	Erro de Relação
Baseline de Desempenho	-	0.53	0.43	0.57
Gemini 2.5 Pro	*Zero-Shot*	**0.98**	0.91	0.91
	3-Shot	0.97	**0.92**	**0.92**
GPT-4o	*Zero-Shot*	0.59	0.45	0.53
	3-Shot	0.66	0.54	0.43
Claude Sonnet 4	*Zero-Shot*	0.81	0.65	0.48
	3-Shot	0.80	0.68	0.53

todas as três categorias. Em contrapartida, tanto o Claude quanto o GPT-4o apresentaram uma dificuldade acentuada em diagnosticar o "Erro de Relação"(F1 de 0.53 e 0.43, respectivamente, em suas melhores versões), que é, possivelmente, o erro mais sutil, pois exige a correta associação contextual entre valores e datas. Ambos foram mais proficientes em identificar o "Erro da Aplicação", a classe majoritária.

Entretanto, um achado notável é o desempenho do *baseline* nessa classe específica ("Erro de Relação"). Com um F1-Score de 0.57, a abordagem baseada em regras superou o GPT-4o e o Claude Sonnet 4, sugerindo que, para a tarefa de associar datas, a lógica programática explícita foi mais eficaz do que a inferência semântica desses dois modelos. Ainda assim, nenhuma abordagem se aproximou da robustez do Gemini, que se provou superior em todos os cenários.

4.2.2. Influência da Complexidade e Agregação do Campo

Para investigar como a complexidade da regra de negócio afeta o diagnóstico, o desempenho foi analisado em dois níveis: primeiro, de acordo com o tipo de composição do campo (Tabela 4) e, segundo, focando especificamente na presença da operação de soma no valor esperado (Tabela 5).

Tabela 4. F1-Score por tipo de composição do campo. Os melhores resultados em cada tipo estão em negrito.

Modelo	Abordagem	Direta	Maior Valor	Soma
Baseline de Desempenho	-	0.51	0.41	0.55
Gemini 2.5 Pro	*Zero-Shot*	**0.94**	0.91	**0.94**
	3-Shot	0.90	**0.93**	**0.94**
GPT-4o	*Zero-Shot*	0.54	0.63	0.43
	3-Shot	0.57	0.51	0.50
Claude Sonnet 4	*Zero-Shot*	0.55	0.73	0.62
	3-Shot	0.55	0.80	0.63

Tabela 5. F1-Score de acordo com a presença da operação de soma na formação do campo.

Modelo	Abordagem	Soma = Não	Soma = Sim
Baseline de Desempenho	-	0.54	0.15
Gemini 2.5 Pro	*Zero-Shot*	**0.93**	**0.95**
	3-Shot	**0.93**	**0.95**
GPT-4o	*Zero-Shot*	0.54	0.25
	3-Shot	0.56	0.34
Claude Sonnet 4	*Zero-Shot*	0.70	0.45
	3-Shot	0.71	0.50

As Tabelas 4 e 5 revelam o impacto da complexidade da tarefa na performance. Mais uma vez, o Gemini se mostrou robusto, mantendo um F1-Score acima de 0.90 independentemente do tipo de composição. Notavelmente, seu desempenho foi ainda melhor nos casos que envolviam soma, indicando uma alta capacidade de agregação. Em contraste, os demais modelos apresentaram um desempenho heterogêneo: o GPT-4o exibiu instabilidade, enquanto o Claude Sonnet 4, de forma inesperada, obteve resultados modestos nos casos de ”Valor Direto”(F1 de 0.55), teoricamente os mais simples.

Contudo, a análise da Tabela 5, que isola a presença da operação de soma no valor esperado, expõe a fragilidade das outras abordagens de forma ainda mais clara, apontando a agregação como o principal divisor de águas. O *baseline* de desempenho demonstra a maior fragilidade, com sua performance despencando para um F1-Score de apenas 0.15. De forma similar, o GPT-4o e o Claude também sofreram quedas de desempenho acentuadas nesta condição, atingindo F1-Scores de apenas 0.34 e 0.50, respectivamente.

Isso indica de forma conclusiva que a capacidade de agregar múltiplas informações para seguir uma regra de negócio complexa é um diferencial chave, onde a abordagem puramente programática falha e a capacidade contextual do Gemini se destaca amplamente.

4.3. Análise de custo

Tabela 6. Tabela de preços das APIs por milhão de tokens (USD), em Julho de 2025.

Modelo	Custo de Entrada	Custo de Saída
Gemini 2.5 Pro	\$1.25* / \$2.50	\$10.00* / \$15.00
GPT-4o	\$2.50	\$10.00
Claude Sonnet 4	\$3.00	\$15.00

** Preço para contextos com menos de 200 mil tokens.*

A análise de custo, quando combinada com os dados de desempenho (Tabela 2) e de preços (Tabela 6), revela um claro trade-off entre acurácia, latência e viabilidade econômica. O Gemini 2.5 Pro se destaca como a solução de melhor custo-benefício: apesar de sua latência ser maior, ele não só oferece a performance mais alta, como também possui o menor custo de entrada base (\$1.25 por milhão de *tokens*).

É notável que, mesmo em sua faixa de preço superior ($2.50), o custo de entrada do Gemini se torna idêntico ao do GPT-4o, que por sua vez apresentou uma performance significativamente inferior. O Claude Sonnet 4 posiciona-se como a opção de maior custo tanto para entrada quanto para saída. Portanto, para a tarefa em questão, o Gemini oferece um desempenho de ponta por um custo operacional menor ou, na pior das hipóteses, equivalente ao dos concorrentes, consolidando-se como a escolha estratégica mais robusta.

4.4. Discussão Geral e Implicações

Os resultados experimentais confirmam de forma conclusiva a eficácia da utilização de LLMs para a tarefa de diagnóstico de erros em sistemas de IDP, com o modelo Gemini 2.5 Pro apresentando um desempenho que o qualifica para implementação em produção. A principal contribuição deste trabalho reside na demonstração de que é possível atingir uma acurácia superior a 90% e, simultaneamente, obter uma redução de mais de 80% no tempo de processamento em comparação com a análise humana.

5. Conclusão e trabalhos futuros

Este trabalho partiu do desafio central de otimizar a manutenção de sistemas de Processamento Inteligente de Documentos (IDP), onde o diagnóstico manual de erros representa um significativo gargalo operacional. O objetivo principal foi, portanto, avaliar a viabilidade e a eficácia de Grandes Modelos de Linguagem (LLMs) para automatizar a tarefa de classificar a origem de falhas em um *pipeline* de IDP. Para isso, foi desenvolvida uma metodologia que incluiu a criação de um *dataset* rotulado a partir de um sistema em produção, uma robusta engenharia de *prompts* baseada em raciocínio algorítmico e um estudo comparativo entre os modelos de fronteira GPT-4o, Claude Sonnet 4 e Gemini 2.5 Pro.

Os resultados experimentais confirmaram de forma conclusiva a hipótese inicial deste trabalho. O modelo Gemini 2.5 Pro alcançou um desempenho de estado da arte, com um F1-Score de 0.94 e um MCC de 0.91, validando a principal contribuição do estudo não apenas pela alta acurácia, mas também pelo ganho de eficiência, obtendo uma redução de 82% no tempo de análise em comparação com o *baseline* temporal. A análise dos resultados sugere que este sucesso está intrinsecamente ligado à capacidade do Gemini de seguir com alta fidelidade as instruções hierárquicas do *prompt*, uma habilidade na qual superou seus concorrentes.

Um dos achados mais significativos foi a notável eficácia da abordagem *zero-shot*. O *prompt* com instruções lógicas detalhadas se mostrou suficiente para extrair a performance máxima dos modelos de ponta, com ganhos apenas marginais na abordagem *3-shot*, não justificando o aumento de custo e latência associado a um *prompt* mais longo. Isso sugere que, para problemas de raciocínio estruturado como este, a clareza e a precisão das instruções podem ser mais impactantes do que a provisão de exemplos.

Apesar dos resultados promissores, é importante reconhecer as limitações deste estudo. O conjunto de dados é limitado em tamanho (140 amostras de teste) e focado em um domínio específico de documentos financeiros. Dessa forma, a generalização dos resultados para outros tipos de documentos ou para um volume massivo de dados deve ser avaliada. Adicionalmente, os modelos foram utilizados como ”caixas-pretas”via API, o que impede uma análise mais profunda de seus mecanismos internos de raciocínio.

As conclusões e limitações deste trabalho abrem diversas possibilidades para pesquisas futuras:

- **Validação em Larga Escala:** Aplicar a metodologia em um conjunto de dados substancialmente maior e mais diversificado, incluindo diferentes tipos de documentos (contratos, notas fiscais, etc.) para verificar a generalização do método.
- **Desenvolvimento de Estratégias Híbridas:** Para otimizar a relação custo-latência em produção, investigar a implementação de um *pipeline* em cascata, onde o *baseline* heurístico ou um modelo mais rápido trata os casos de alta confiança, e o Gemini é acionado apenas para os diagnósticos mais ambíguos.
- **Exploração de Modelos de Código Aberto:** Investigar o desempenho de modelos de linguagem de código aberto (como Llama 3 ou outros) na mesma tarefa. A aplicação de técnicas de *fine-tuning* nesses modelos poderia, potencialmente, alcançar um desempenho similar ao das APIs proprietárias, com a vantagem de maior controle, menor custo operacional e latência reduzida. Para viabilizar a criação de um conjunto de dados maior, necessário para o *fine-tuning*, a solução proposta neste trabalho poderia ser utilizada como uma ferramenta de rotulagem semi-automatizada, gerando um *dataset* inicial que seria posteriormente validado por um especialista humano, otimizando significativamente o processo de anotação manual.

Em suma, este trabalho demonstrou que a aplicação de LLMs, guiada por uma engenharia de *prompts* sofisticada, é uma solução poderosa e prática para automatizar o diagnóstico de erros em sistemas de IDP. A metodologia proposta não apenas estabelece um novo patamar de eficiência para a manutenção desses sistemas, mas também contribui com *insights* sobre como estruturar a interação entre LLMs e tarefas de raciocínio lógico.

Referências

[1] Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020a). LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1192-1200). ACM.

[2] Garncarek, Ł., Powalski, R., Starczewski, B., Graliński, F., Wrembel, R., Goc, M., & Stanisławek, T. (2020). LAMBERT: Layout-Aware language model for information extraction. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 1913-1922). IEEE.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems, 30* (pp. 5998-6008). Curran Associates, Inc.

[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems, 33* (pp. 1877–1901). Curran Associates, Inc.

[5] Zheng, L., Chiang, W. L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., et al. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*.

[6] Microsoft Corporation. (2025). *Azure AI Vision - Document Analysis, Read API v3.2*. https://learn.microsoft.com/pt-br/azure/ai-services/document-intelligence/concept-read. Acessado em 22 de julho de 2025.

[7] OpenAI. (2024). *Hello GPT-4o*. https://openai.com/index/hello-gpt-4o/. Acessado em 22 de julho de 2025.

[8] Anthropic. (2025). *Introducing Claude 4*. https://www.anthropic.com/news/claude-4. Acessado em 22 de julho de 2025.

[9] Google. (2025). *Gemini 2.5: Our most intelligent AI model*. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking. Acessado em 22 de julho de 2025.

[10] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems, 35* (pp. 24824-24837). Curran Associates, Inc.

[11] Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics, 21*(1), 1-13.