Journal of Health Informatics

J Health Inform. 2026;18:e1788
DOI: https://doi.org/10.59681/2175-4411.v18.iEspecial.2026.1788

ARTIGO COMPLETO
CBIS 2026

Arquitetura multiagente para processamento e codificação (CID-10) automatizada de notas clínicas

Multi-agent architecture for automated processing and coding (ICD-10) of clinical notes

Gabriela Carolina Ferreira Gimenez[1], Clauirton de Albuquerque Siebra[1], Luiz Fernando Fonsêca Pinheiro de Lima[1]

ORCID: 0009-0009-8145-2837 · 0000-0002-9742-2175 · 0000-0003-1992-6316

[1] Universidade Federal da Paraíba, João Pessoa (PB), Brasil.

FICHA DO ARTIGO

TIPO
Artigo completo · CBIS 2026

IDENTIFICADOR
e1788

DOI
10.59681/2175-4411.v18.iEspecial.2026.1788

RECEBIDO
17/05/2026

APROVADO
20/07/2026

PROCESSO DE AVALIAÇÃO
Double Blind Review

EDITOR RESPONSÁVEL
Juliano de Souza Gaspar

ACESSO
Aberto, CC BY 4.0

AUTOR CORRESPONDENTE
Gabriela Carolina Ferreira Gimenez
gimenez.g.gabriela@gmail.com

LICENÇA

Este é um artigo publicado em acesso aberto sob a licença Creative Commons CC BY 4.0, que permite uso, distribuição e reprodução em qualquer meio, desde que o trabalho original seja corretamente citado.

CC BY 4.0

RESUMO

Objetivo: Este trabalho propõe uma arquitetura multiagente baseada em microsserviços para automatizar a codificação CID-10 de notas clínicas em português brasileiro. **Método:** A solução integra três agentes especializados (extração de entidades, normalização terminológica e codificação), orquestrados via LangGraph com suporte a processamento em lote, execução concorrente e cache Redis. As etapas de extração e normalização utilizam grandes modelos de linguagem; a codificação combina heurísticas de busca textual com consultas à API NIH Clinical Tables. O pipeline foi validado sobre 119 notas sintéticas. **Resultados:** O pipeline demonstrou alta cobertura operacional (99,16% de conclusão), média de 3,77 entidades extraídas por nota e 12,24 candidatos CID-10 retornados, com rastreabilidade estruturada em todas as etapas. Persistem limitações relacionadas à dependência de recursos em inglês e à ausência de avaliação com dados reais. **Conclusão:** A abordagem é promissora, recomendando-se evoluções com modelos especializados, ajuste fino e geração aumentada por recuperação (RAG) para incremento de acurácia.

Descritores: Classificação Internacional de Doenças; arquitetura multiagente; processamento de linguagem natural.

ABSTRACT

Objective: This work proposes a microservices-based multi-agent architecture to automate ICD-10 coding of clinical notes in Brazilian Portuguese. **Method:** The solution integrates three specialized agents (entity extraction, terminological normalization, and coding) orchestrated via LangGraph with support for batch processing, concurrent execution, and Redis caching. The extraction and normalization stages use large language models; the coding stage combines text-based search heuristics with queries to the NIH Clinical Tables API. The pipeline was validated on 119 synthetic notes. **Results:** The pipeline demonstrated high operational coverage (99.16% completion rate), an average of 3.77 entities extracted per note, and 12.24 ICD-10 candidates returned, with structured traceability across all stages. Limitations remain regarding dependence on English-language resources and the absence of evaluation with real clinical data. **Conclusion:** The approach is promising, with further developments recommended, including specialized models, fine-tuning, and retrieval-augmented generation (RAG) to improve accuracy.

Keywords: International Classification of Diseases; multi-agent architecture; natural language processing.

CBIS 2026 XXI CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE

Artigo apresentado no XXI Congresso Brasileiro de Informática em Saúde (CBIS 2026), realizado de 23 a 25 de setembro de 2026, em Brasília (DF), e publicado na seção CBIS 2026 do Journal of Health Informatics.

COMO CITAR ESTE ARTIGO: Gimenez GCF, Siebra CA, Lima LFFP. Arquitetura multiagente para processamento e codificação (CID-10) automatizada de notas clínicas. J Health Inform 2026;18:e1788. doi: https://doi.org/10.59681/2175-4411.v18.iEspecial.2026.1788

1 INTRODUÇÃO

A codificação clínica é fundamental para processos de faturamento, vigilância epidemiológica e tomada de decisão em saúde. A Classificação Internacional de Doenças (CID), mantida pela Organização Mundial da Saúde, é o principal padrão para esse fim, organizando diagnósticos e procedimentos em códigos alfanuméricos. Na versão CID-10, o conjunto já é extenso, com dezenas de milhares de códigos diagnósticos; a CID-11, lançada em 2022, expande ainda mais esse universo, com mais de 120 mil termos modificáveis e ferramentas digitais para integração com sistemas de informação em saúde [1].

Historicamente, a atribuição desses códigos depende de codificadores humanos que leem notas clínicas, interpretam sinais, sintomas e diagnósticos e buscam manualmente os códigos nos manuais. Esse processo é cognitivo e demorado: estimativas internacionais apontam de 7 a 8 minutos por caso em cenários de alto volume, e até 30 minutos em casos complexos, criando gargalos em faturamento e análise de dados. Além disso, está sujeito a erros e inconsistências entre profissionais, com impacto sobre reembolso, qualidade da informação e condutas clínicas [2,3].

Nesse contexto, a automatização da codificação em CID tornou-se tema central em processamento de linguagem natural aplicado à saúde. A tarefa é modelada como um problema de classificação multirrótulo em larga escala, pois uma mesma nota pode receber vários códigos, sobre textos heterogêneos repletos de abreviações, erros de digitação e linguagem coloquial. Revisões recentes mostram avanços com modelos baseados em Transformers e variantes biomédicas, como BERT e RoBERTa, incluindo abordagens de pareamento semântico profundo entre texto e descrições de códigos [2-4].

Os grandes modelos de linguagem (LLMs) também passaram a ser explorados para essa tarefa. Modelos como o GPT-4 capturam bem o vocabulário clínico e atingem altos valores de *recall*, mas com baixa precisão e risco de alucinações e códigos irrelevantes [5,6]. Para mitigar essas limitações, diferentes arquiteturas foram propostas: um assistente de codificação que combina RoBERTa para extração de termos-chave com GPT-4 e recuperação por descrição de códigos, aproximando o fluxo do trabalho humano [6]; o *framework* "Code Like Human" (CLH), que decompõe a tarefa em quatro agentes especializados cobrindo toda a CID-10 com mais de 70 mil códigos [3]; e um sistema multiagente com cinco papéis (paciente, médico, codificador, revisor e ajustador) que integra a estrutura de prontuários em subjetivo, objetivo, avaliação e plano (SOAP) e obtém ganhos sobre estratégias de *chain-of-thought* para códigos comuns e raros [5].

Do ponto de vista conceitual, agentes autônomos são definidos classicamente como sistemas situados em um ambiente, capazes de percebê-lo e agir em função de objetivos próprios [7]. No contexto atual dos LLMs, esse conceito evoluiu para arquiteturas em que o modelo atua como núcleo de decisão, planejamento e execução, apoiado por módulos de perfil, memória e ação [8,9]. Em sistemas multiagentes baseados em LLMs, o foco desloca-se da autonomia plena de cada componente para a especialização de papéis, a comunicação entre agentes e a resolução coletiva de tarefas. É nesse sentido contemporâneo que o termo é empregado neste trabalho.

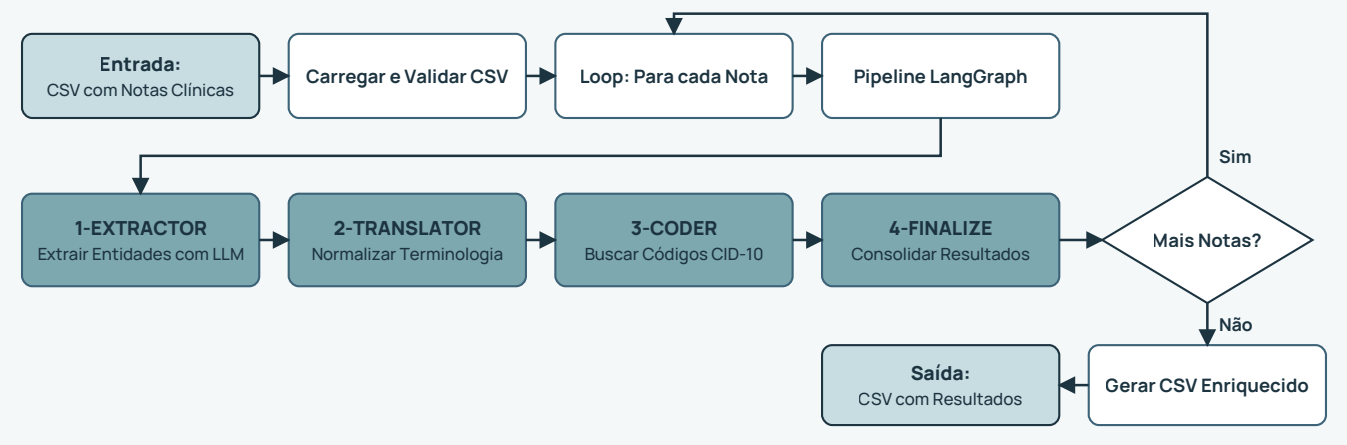
Embora esses avanços sejam expressivos, os estudos são majoritariamente avaliados em bases em inglês, como MIMIC-III [10], ou espanhol, sem tratar especificamente o português brasileiro. Além disso, grande parte das propostas foca em acurácia de modelos, com menor atenção a aspectos de engenharia de sistema, como processamento em lote, escalabilidade em microsserviços, integração com APIs de CID-10 e rastreabilidade do *pipeline* [2,4-6].

Este trabalho insere-se nesse cenário com foco no português brasileiro e na perspectiva de sistema. O objetivo é desenvolver uma arquitetura multiagente baseada em microsserviços para normalização de notas clínicas e codificação assistida em CID-10, com decomposição explícita das etapas de extração de entidades clínicas, normalização terminológica e recuperação de candidatos a códigos via API NIH Clinical Tables [11], orquestrada por LangGraph com cache Redis e geração de saídas auditáveis em CSV. A abordagem trata a inteligência artificial (IA) não como substituto do codificador humano, mas como apoio estruturado, transparente e escalável à sua prática.

2 MÉTODOS

A seção de métodos descreve a arquitetura proposta e os componentes que compõem o sistema, detalhando o processador em lote responsável pela orquestração do *pipeline* e os três agentes especializados, extração, normalização e codificação. São apresentados o modelo de execução baseado em grafo de estados via LangGraph, o mecanismo de processamento paralelo com concorrência configurável e a comunicação entre serviços por meio de APIs REST. A seção também detalha as estratégias adotadas em cada agente, incluindo o uso de modelos de linguagem para extração e normalização, heurísticas e consultas à API NIH Clinical Tables na etapa de codificação, bem como o uso de cache Redis. Por fim, são descritos o fluxo completo de processamento das notas clínicas, a estrutura dos dados de entrada e saída e os mecanismos de tratamento de erros e garantia de robustez do sistema.

Figura 1. Diagrama do Fluxo Principal



Fonte: Elaborado pelos autores (2026).

2.1 Arquitetura do Sistema

O sistema desenvolvido segue uma arquitetura multiagente baseada em microsserviços, projetada para processamento em lote de notas clínicas em português brasileiro, conforme a Figura 1. A arquitetura é composta por quatro componentes principais: um processador em lote (Batch Processor) que atua como orquestrador central, e três agentes especializados que executam tarefas específicas do *pipeline* de codificação: extração de entidades clínicas, normalização terminológica e consulta de códigos CID-10.

A orquestração do *pipeline* é feita pelo Batch Processor por meio do LangGraph, que implementa um grafo de estados com execução sequencial dos nós (extrator, tradutor, codificador e finalização). O processamento de múltiplas notas ocorre em paralelo, com concorrência limitada e configurável (por padrão, até cinco notas simultâneas por *job*), permitindo maior *throughput* sem sobrecarregar as APIs externas.

A comunicação entre componentes é feita por APIs REST assíncronas, com FastAPI como *framework* HTTP. Cada agente é implementado como um microsserviço independente e containerizado com Docker, permitindo escalabilidade horizontal e manutenção isolada. O agente de codificação utiliza Redis como camada de cache para códigos CID-10 consultados, reduzindo chamadas à API NIH Clinical Tables e a latência em operações de codificação.

2.2 Processador em Lote (Batch Processor)

O Batch Processor é responsável pela orquestração do *pipeline* de processamento das notas clínicas. Ele recebe arquivos CSV contendo a coluna nota_clinica, cria e gerencia *jobs* de processamento, coordena a execução sequencial das etapas via LangGraph (extrator, tradutor, codificador e finalização) e registra o status de cada nota processada.

O componente também realiza a persistência dos resultados e metadados em banco SQLite, utilizando SQLAlchemy para representar os *jobs* e os registros individuais de cada nota. Ao final do processamento, gera um CSV enriquecido contendo o texto original e o resultado estruturado em JSON. A execução é assíncrona, com suporte a processamento paralelo e concorrência configurável, permitindo o processamento simultâneo de múltiplas notas por *job*.

2.3 Agente Extrator (Extractor Agent)

O Extractor Agent é responsável pela identificação de entidades clínicas relevantes nas notas, tarefa modelada como *named entity recognition* (NER). Foram consideradas quatro categorias: *disease* (doenças), *symptom* (sintomas), *sign* (sinais clínicos) e *anatomy* (anatomia).

A extração é realizada por meio de modelo de linguagem configurável via OPENAI_MODEL, orientado por *prompt* externo em YAML. As respostas são submetidas a pós-processamento determinístico para extração do JSON, validação das categorias e ajuste dos *spans* quando necessário. A saída é validada por *schema* Pydantic e inclui, para cada entidade, o trecho original, a categoria, a posição no texto e a fonte de extração. O *endpoint* /extract recebe a nota clínica e retorna a lista de entidades extraídas, acompanhada de metadados como modelo utilizado, método de extração e contagem de entidades.

2.4 Agente Tradutor (Translator Agent)

O Translator Agent realiza a normalização terminológica das entidades extraídas, convertendo termos leigos ou populares em terminologia médica padronizada em português brasileiro e gerando o termo equivalente em inglês para consulta em bases internacionais de CID-10.

A normalização é realizada por modelo de linguagem configurável via OPENAI_MODEL, orientado por *prompt* externo em YAML. O agente prioriza nomes oficiais da CID-10 quando há correspondência clara; caso contrário, utiliza terminologia médica técnica. Para cada entidade recebida, retorna o termo original, o

termo normalizado, o equivalente em inglês, a categoria herdada da extração e a fonte do método utilizado.

O *endpoint* /translate recebe a lista de entidades do Extractor Agent e retorna uma lista de traduções normalizadas, mantendo correspondência com as entidades de entrada. Os metadados incluem modelo utilizado, versão do *prompt* e contagem de traduções.

2.5 Agente Codificador (Coder Agent)

O Coder Agent recupera candidatos a códigos CID-10-CM a partir dos termos normalizados pelo Translator Agent. Para cada entidade, o termo de consulta é definido de forma hierárquica, priorizando *english_term*, seguido de *normalized_text* e, por fim, *original_text*.

O processo de codificação combina cache Redis e consultas assíncronas à API NIH Clinical Tables. Inicialmente, o agente verifica se o termo já possui resultado armazenado em cache; caso contrário, realiza requisições à API com parâmetros configurados para retornar até cinco candidatos por termo. Quando necessário, a consulta pode ser reformulada por meio de variantes textuais, como remoção de termos vazios ou equivalências PT-EN auxiliares.

Quando nenhuma correspondência é encontrada, o agente não atribui códigos manualmente. Em vez disso, retorna uma entrada estruturada com indicação explícita de ausência de correspondência (*source="no_match"*), preservando a rastreabilidade do processo e reduzindo o risco de geração de códigos não fundamentados. O *endpoint* /code recebe os termos normalizados e retorna os candidatos CID-10-CM, acompanhados de metadados como acertos de cache, chamadas à API e indicador *no_match_found*.

2.6 Pipeline de Processamento

O *pipeline* de processamento é orquestrado pelo Batch Processor mediante LangGraph, com um grafo de estados em quatro nós principais (extrator, tradutor, codificador e finalização), conforme ilustrado na Figura 1. Múltiplas notas são processadas em paralelo por *job*, com concorrência limitada e configurável. Para cada nota clínica no arquivo CSV de entrada:

Etapa 1 (extração): o Batch Processor envia o texto da nota clínica ao Extractor Agent via requisição POST assíncrona. O agente identifica entidades clínicas e retorna uma lista estruturada em JSON com categorias (*disease*, *symptom*, *sign*, *anatomy*), *spans* (posições no texto) e identificador da fonte de extração.

Etapa 2 (normalização): as entidades extraídas são enviadas ao Translator Agent, que converte termos leigos em terminologia técnica em português brasileiro e gera o termo em inglês (*english_term*) para busca. O agente retorna uma lista de traduções normalizadas, mantendo correspondência com as entidades originais.

Etapa 3 (codificação): os termos normalizados são enviados ao Coder Agent, que consulta códigos CID-10 por cache Redis e API NIH Clinical Tables; quando nenhuma correspondência é encontrada, o serviço retorna uma entrada estruturada de ausência de código. O agente retorna uma lista de códigos com descrições e metadados (termo consultado, estratégia de busca).

Etapa 4 (finalização): o nó de finalização consolida os resultados das três etapas em um objeto JSON estruturado. Em caso de erro em qualquer etapa, o *pipeline* segue para o nó de finalização com o estado disponível, preservando os resultados das etapas já concluídas; etapas com falha resultam em blocos com listas vazias no objeto final.

Os experimentos foram executados com concorrência configurada para o valor padrão de cinco notas simultâneas por *job*. Após o processamento de cada nota (que pode ocorrer em paralelo com outras), os resultados são armazenados no banco de dados e os contadores do *job* são atualizados. Ao final do processamento de todas as notas do *job*, um arquivo CSV enriquecido é gerado contendo as colunas nota_clinica (texto original) e resultado (JSON com entidades extraídas, normalizadas e códigos CID-10).

Para a validação funcional desta etapa, foi utilizado um conjunto de aproximadamente 1.000 notas clínicas sintéticas em português brasileiro, geradas com apoio de modelos de linguagem sob critérios definidos pelos autores. Essa escolha foi deliberada e orientada pelo objetivo desta fase do trabalho: verificar o comportamento arquitetural do *pipeline* ponta a ponta, desde a ingestão das notas até a geração das saídas estruturadas em JSON e CSV, sem que variáveis externas, como ruído real, inconsistências de digitação ou variabilidade clínica não controlada, interferissem na análise do fluxo de processamento. O uso de dados sintéticos permite isolar o comportamento de cada etapa do *pipeline* e avaliar a rastreabilidade das saídas de forma controlada, o que seria mais difícil com dados clínicos reais heterogêneos nesta fase inicial. A construção do corpus buscou simular variações de idade, sexo, contexto assistencial, sinais, sintomas, antecedentes clínicos e diferentes níveis de especificidade terminológica, incluindo expressões mais diretas, descrições ambíguas e formulações potencialmente ruidosas. Todo o conjunto foi revisado pelos autores, que verificaram a coerência textual de cada nota e sua adequação aos critérios definidos para os testes funcionais, rejeitando ou corrigindo registros com inconsistências clínicas evidentes ou estrutura inadequada. A avaliação com dados clínicos reais ou anotados manualmente constitui etapa prevista para trabalhos futuros.

2.7 Tratamento de Erros e Resiliência

O sistema adota mecanismos de validação e tratamento de falhas em múltiplas camadas. Nos agentes, a validação de entrada é realizada com Pydantic. No Batch Processor, falhas de rede, *timeout* ou respostas inválidas são capturadas no estado do *pipeline*, permitindo o encaminhamento ao nó de finalização sem interromper o processamento do lote.

No Translator Agent, saídas incompletas são preenchidas com o texto original, preservando a correspondência com a entrada. No Coder Agent, a ausência de candidatos na API NIH é tratada como condição de não correspondência, retornando-se uma entrada explícita do tipo *no_match*.

A persistência em SQLite registra, por nota, o texto original, o status de processamento e, quando aplicável, a mensagem de erro. Falhas não tratadas no orquestrador tendem a marcar a nota como falha, enquanto falhas absorvidas no interior do grafo podem resultar em saídas parciais ainda contabilizadas como processadas. Ao final, são atualizados os contadores agregados do *job*.

3 RESULTADOS

Os experimentos desta fase evidenciaram a viabilidade funcional do *pipeline* multiagente proposto para processamento de notas clínicas em português brasileiro, contemplando as etapas de extração de entidades, normalização terminológica e recuperação de candidatos a códigos CID-10. Em todas as amostras analisadas, o sistema gerou saídas estruturadas organizadas em três blocos principais, *extractor*, *translator* e *coder*, além de metadados operacionais relacionados à execução, como número de entidades identificadas, chamadas à API, ocorrências de cache e indicação de ausência de correspondência na etapa de codificação.

De modo geral, observou-se boa coerência nos casos em que a nota apresentava sinais e sintomas mais diretos. Na nota com palpitações e edema em membros inferiores, por exemplo, o *pipeline* preservou adequadamente o encadeamento entre extração, normalização e codificação, retornando R00.2 para palpitações e múltiplos candidatos plausíveis para edema, como R60.0, R60.1 e R60.9. Resultado semelhante foi observado na nota com dispneia progressiva e fibrilação atrial, na qual a etapa de codificação retornou R06.00 para *Dyspnea*, indicando correspondência plausível entre o sintoma clínico e os candidatos CID-10 recuperados.

Também foram observadas saídas globalmente coerentes em notas respiratórias e otorrinolaringológicas. Na nota com náuseas, vômitos e sibilos expiratórios difusos, o sistema sugeriu R06.2 para *Wheezing* e opções compatíveis para o grupo de náusea e vômitos, como R11.0 e R11.2. De modo análogo, na nota pediátrica com otalgia e sensação de ouvido tapado, o *pipeline* retornou candidatos adequados para otalgia, mostrando que a arquitetura consegue produzir resultados clinicamente plausíveis em diferentes perfis de nota.

Além da coerência semântica parcial observada nesses exemplos, os resultados também indicaram aspectos positivos do ponto de vista operacional. Em diversos registros, o campo *cache_hits* aparece preenchido, e algumas respostas são marcadas como “retornado via cache”, sugerindo reaproveitamento de consultas anteriores e potencial redução de chamadas redundantes à API externa. Esses achados reforçam que a arquitetura proposta é viável como prova de conceito funcional, tanto pela geração de artefatos intermediários estruturados quanto pela capacidade de manter encadeamento entre extração, normalização e busca de códigos.

Por outro lado, também foram observados erros semânticos relevantes. Em uma nota com dor articular migratória, o extrator identificou “unidade básica de saúde” como entidade anatômica, o que levou o codificador a retornar candidatos sem aderência clínica, como E21.0 (hiperparatireoidismo primário). Esse caso evidencia que a arquitetura ainda não filtra adequadamente elementos contextuais e administrativos que não deveriam seguir para a etapa de codificação.

Embora os resultados sustentem a viabilidade funcional da arquitetura, a qualidade semântica das decisões intermediárias ainda demanda refinamentos, especialmente na filtragem de entidades contextuais, na normalização de expressões ambíguas e na desambiguação dos candidatos retornados.

Os exemplos apresentados nesta seção são ilustrativos do comportamento observado ao longo do *pipeline*; a quantificação sistemática das métricas de cobertura, extração e codificação é apresentada na seção seguinte.

A Tabela 1 apresenta uma seleção de casos representativos do comportamento observado ao longo do *pipeline*, contemplando notas com diferentes perfis clínicos e variados graus de correspondência entre as entidades extraídas e os candidatos a códigos CID-10 retornados. Os exemplos foram selecionados de forma a ilustrar tanto situações de boa aderência semântica quanto casos em que a extração de entidades contextuais comprometeu a qualidade da codificação, evidenciando os principais padrões de acerto e limitação identificados no subconjunto analisado.

Tabela 1. Exemplos representativos de viabilidade funcional do pipeline

Nota clínica	Entidades/normalização	Resultado obtido	Interpretação
Paciente idoso de 65 anos, sexo masculino, relata palpitações em pior progressão há 48h, nega outros sintomas no relato. Antecedentes: idoso ativo, mora sozinho. Mmii edemaciados 2+/4+. Encaminhamento para especialista indicado.	"palpitações" → *Palpitations*; "Mmii" → *Lower limbs*; "edemaciados" → *Edema*	R00.2 para palpitações; edema com múltiplos candidatos (R60.0, R60.1, R60.9)	Exemplo de boa correspondência para sintoma direto, mas com ambiguidade residual em sinais inespecíficos
Paciente de 47 anos, sexo masculino, comparece à unidade básica de saúde com cefaleia holocraniana pulsátil há 1 semana. Antecedentes: portador de DM2 com uso de metformina. Hipotensão postural referida. Conduta: observação breve e alta com critérios de alarme.	"cefaleia" → *Headache*; "holocraniana" → *Skull*; "DM2" → *Type 2 diabetes mellitus*; "hipotensão postural" → *Orthostatic hypotension*	R51.9 para cefaleia; I95.1 para hipotensão ortostática	Núcleo clínico preservado, mas com extração excessiva de componente anatômico irrelevante
Paciente idoso de 94 anos, sexo feminino, idoso ativo, mora sozinho. Principal queixa: náuseas e vômitos em pior progressão há 48h. Exame físico: sibilos expiratórios difusos. Plano: quadro compatível com exacerbação de base crônica.	"náuseas/vômitos" → *Nausea and vomiting*; "sibilos" → *Wheezing*	Grupo R11 para náusea/vômito e R06.2 para *wheezing*	Saída globalmente coerente para sintomas e sinais respiratórios

Fonte: Elaborado pelos autores (2026).

Tabela 2. Principais indicadores quantitativos do processamento

Indicador	Resultado
Notas clínicas analisadas	119 de 1.000
Conclusão do *pipeline* completo	99,16%
Entidades extraídas por nota	Média de 3,77
Traduções normalizadas por nota	Média de 3,74
Códigos CID-10 retornados por nota	Média de 12,24

Fonte: Elaborado pelos autores (2026).

4 RESULTADOS QUANTITATIVOS DO PROCESSAMENTO

A consolidação dos resultados estruturados gerados pelo *pipeline* foi realizada sobre um subconjunto de 119 notas clínicas, correspondente às execuções concluídas antes do esgotamento da cota de chamadas à API externa. Do total de 1.000 notas presentes no arquivo de entrada, esses 119 registros foram efetivamente analisados, todos produzindo saídas em formato JSON válido contendo os blocos *"extractor"*, *"translator"* e *"coder"*, além de metadados detalhados por etapa.

Em termos de cobertura do *pipeline*, os resultados demonstraram alto grau de completude dentro do subconjunto válido. A etapa de extração apresentou ativação em 100% das notas analisadas, enquanto as etapas de tradução e codificação foram concluídas com sucesso em 99,16% dos casos. Apenas uma nota completou a extração, mas falhou nas etapas subsequentes, correspondendo ao único caso de *pipeline* incompleto no subconjunto. Esse padrão evidencia que, em condições normais de operação da API, o *pipeline* executa o fluxo completo de forma consistente.

Em termos de extração de entidades, foram identificadas em média 3,77 entidades por nota clínica, com desvio-padrão de 1,05, variando entre 2 e 6 entidades por nota. As categorias mais frequentes foram "*symptom*" (132 ocorrências), "*disease*" (127) e "*sign*" (123), seguidas de "*anatomy*" (63), indicando que o *pipeline* captura predominantemente manifestações clínicas, diagnósticos e achados de exame físico presentes nas notas.

A etapa de tradução apresentou correspondência quase total com a extração, com média de 3,74 traduções por nota e correlação de Spearman de 0,98 com o número de entidades extraídas. A razão entre traduções e entidades atingiu mediana de 1,0, confirmando que, nas execuções bem-sucedidas, praticamente todas as entidades extraídas foram efetivamente normalizadas e traduzidas para inglês.

Na etapa de codificação, foram retornados em média 12,24 códigos CID-10 por nota, com desvio-padrão de 5,71 e máximo de 29 códigos em uma única nota. O número de códigos únicos por nota seguiu distribuição semelhante, com média de 11,98 e mediana de 11. A correlação entre número de traduções e número de códigos CID foi de 0,65, indicando associação moderada entre a quantidade de termos normalizados e a quantidade de candidatos de codificação retornados. Os códigos mais frequentemente retornados incluíram R60.9 (edema não especificado, 17 ocorrências), variantes do grupo G57 (mononeuropatias dos membros inferiores, 14 ocorrências cada) e I10 (hipertensão essencial, 11 ocorrências).

Em relação ao tratamento de falhas e à rastreabilidade, os metadados do *pipeline* registraram, para cada etapa, campos estruturados como *"entities_count"*, *"extraction_method"* e *"model"* no extrator; *"translations_count"*, *"gpt_translations"* e *"model"* no tradutor; e *"api_calls"*, *"cache_hits"* e *"no_match_found"* no codificador. Esse conjunto de metadados, associado à taxa de conclusão de 99,16% observada no subconjunto válido, reforça que o sistema executa corretamente o fluxo completo,

da ingestão à geração de saídas estruturadas, com preservação dos artefatos intermediários ao longo do processamento, sendo a limitação de cota da API externa o principal fator condicionante da cobertura total do experimento.

5 CONSIDERAÇÕES FINAIS

A análise quantitativa foi realizada sobre um subconjunto de 119 notas clínicas, correspondente às execuções concluídas antes do esgotamento da cota da API externa, do total de 1.000 notas no arquivo de entrada. Todos os 119 registros produziram saídas JSON válidas contendo os blocos extractor, translator e coder, com metadados detalhados por etapa.

A cobertura do *pipeline* foi alta: a extração ativou em 100% dos casos, e as etapas de tradução e codificação foram concluídas com sucesso em 99,16%, com apenas uma nota apresentando *pipeline* incompleto. Na extração, foram identificadas em média 3,77 entidades por nota (desvio-padrão = 1,05; variação de 2 a 6), predominando as categorias *symptom* (132 ocorrências), *disease* (127) e *sign* (123), seguidas de *anatomy* (63). A tradução acompanhou a extração de forma quase integral, com média de 3,74 traduções por nota e correlação de Spearman de 0,98, confirmando que praticamente todas as entidades extraídas foram normalizadas e traduzidas para o inglês.

Na codificação, foram retornados em média 12,24 códigos CID-10 por nota (desvio-padrão = 5,71; máximo de 29), com correlação moderada de 0,65 entre o número de traduções e de candidatos retornados. Os códigos mais frequentes foram R60.9 (edema não especificado, 17 ocorrências), variantes do grupo G57 (mononeuropatias dos membros inferiores, 14 ocorrências cada) e I10 (hipertensão essencial, 11 ocorrências).

Os metadados estruturados registrados em cada etapa (como entities_count e extraction_method no extrator; translations_count e gpt_translations no tradutor; e api_calls, cache_hits e no_match_found no codificador) reforçam a rastreabilidade ponta a ponta do sistema, sendo a limitação de cota da API externa o principal fator condicionante da cobertura total do experimento.

Por fim, destaca-se que, devido ao *pipeline* proposto para a geração de dados sintéticos ser composto por diferentes etapas utilizando IA generativa, estes dados podem apresentar uma organização e distribuição diferente do encontrado em contextos reais no português brasileiro, já que as notas sintéticas têm linguagem mais regular, mais limpa e mais alinhada ao comportamento esperado dos modelos. Deste modo, ainda é relevante a estruturação de um conjunto de dados de notas clínicas reais neste idioma.

6 ASPECTOS ÉTICOS

Este trabalho não envolveu coleta ou uso de dados de seres humanos, sendo baseado exclusivamente em dados sintéticos e/ou públicos, não havendo necessidade de submissão a Comitê de Ética em Pesquisa. O sistema proposto não realiza identificação de indivíduos nem tratamento de dados sensíveis reais.

Contribuição dos autores. Os autores contribuíram igualmente em todas as etapas de construção do artigo: concepção e delineamento do estudo; coleta, análise e interpretação dos dados; redação e revisão crítica do manuscrito; e aprovação da versão final.

Financiamento. Para este estudo não houve fonte de financiamento.

Disponibilidade de dados e materiais. Os autores não declararam a disponibilidade dos dados e materiais da pesquisa.

Conflitos de interesse. Os autores declaram que não há conflito de interesses de ordem financeira, comercial, política, acadêmica ou pessoal relacionado ao conteúdo deste estudo.

Declaração de uso de IA. Foram utilizadas ferramentas de Inteligência Artificial, incluindo ChatGPT, como apoio à geração da base de dados sintética empregada na validação funcional do *pipeline*, bem como à revisão linguística, organização do texto e esclarecimento de dúvidas ao longo do desenvolvimento do trabalho. O uso dessas ferramentas na geração dos dados sintéticos ocorreu sob definição de critérios pelos autores e com posterior revisão crítica do material produzido. As ferramentas não foram utilizadas de forma autônoma para definição das conclusões do estudo. Todo o conteúdo foi revisado criticamente pelos autores, que são integralmente responsáveis pelo conteúdo final.

REFERÊNCIAS

1. World Health Organization. ICD-11 2022 release: key features [Internet]. 2022 Feb 11 [cited 2026 Apr 7]. Available from: https://www.who.int/news/item/11-02-2022-icd-11-2022-release
2. Dong H, Falis M, Whiteley W, Alex B, Matterson J, Ji S, et al. Automated clinical coding: what, why, and where we are? NPJ Digit Med. 2022;5(1):159. doi: https://doi.org/10.1038/s41746-022-00705-7
3. Motzfeldt AG, Edin J, Christensen CL, Hardmeier C, Maaløe L, Rogers A. Code Like Humans: A Multi-Agent Solution for Medical Coding. In: Findings of the Association for Computational Linguistics: EMNLP 2025. Stroudsburg, PA, USA: Association for Computational Linguistics; 2025. p. 22612-27. doi: https://doi.org/10.18653/v1/2025.findings-emnlp.1231
4. Chen Y, Chen H, Lu X, Duan H, He S, An J. Automatic ICD-10 coding: Deep semantic matching based on analogical reasoning. Heliyon. 2023;9(4):e15570. doi: https://doi.org/10.1016/j.heliyon.2023.e15570

5. Li R, Wang X, Yu H. Exploring LLM multi-agents for ICD coding. arXiv [Preprint]. 2024:arXiv:2406.15363.

6. Puts S, Zegers CML, Dekker A, Bermejo I. Developing an ICD-10 Coding Assistant: Pilot Study Using RoBERTa and GPT-4 for Term Extraction and Description-Based Code Selection. JMIR Form Res. 2025;9:e60095. doi: https://doi.org/10.2196/60095

7. Franklin S, Graesser A. Is It an agent, or just a program?: A taxonomy for autonomous agents. In: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg; 1997. p. 21-35. doi: https://doi.org/10.1007/bfb0013570

8. Guo T, Chen X, Wang Y, Chang R, Pei S, Chawla NV, et al. Large Language Model Based Multi-agents: A Survey of Progress and Challenges. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence. California: International Joint Conferences on Artificial Intelligence Organization; 2024. p. 8048-57. doi: https://doi.org/10.24963/ijcai.2024/890

9. Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, et al. A survey on large language model based autonomous agents. Front. Comput. Sci. 2024;18(6):186345. doi: https://doi.org/10.1007/s11704-024-40231-1

10. Johnson AE, Pollard TJ, Shen L, Lehman LW, Feng M, Ghassemi M, et al. MIMIC-III, a freely accessible critical care database. Sci Data. 2016;3:160035. doi: https://doi.org/10.1038/sdata.2016.35

11. National Library of Medicine. Clinical Table Search Service: API for ICD-10-CM [Internet]. 2025 [cited 2026 Apr 7]. Available from: https://clinicaltables.nlm.nih.gov/apidoc/icd10cm/v3/doc.html

NOTAS EDITORIAIS

O Journal of Health Informatics é uma publicação científica mantida pela Sociedade Brasileira de Informática em Saúde, associação científica sem fins lucrativos dedicada ao desenvolvimento da Informática em Saúde e da Saúde Digital no Brasil. O processo editorial é conduzido com independência científica, sem interferência institucional, comercial ou de patrocinadores, e segue as boas práticas internacionais de ética em publicação científica.

Este artigo foi avaliado por pares, em sistema duplo-cego, por especialistas da área. A equipe editorial usou ferramentas de inteligência artificial, com supervisão humana, na triagem inicial, na conferência do texto, das referências e dos números e, quando necessário, no redesenho de figuras, quadros e tabelas, aprovado pelos autores na prova. As ferramentas e as salvaguardas estão na Política de uso de inteligência artificial da revista. As decisões editoriais foram tomadas por pessoas.

SBIS Sociedade Brasileira de Informática em Saúde