Abordagem de Migração de Dados Não Estruturados para Bancos de Dados Relacionais Usando Modelos de Linguagem de Larga Escala Pré Treinados

Leon J. M. Ferreira[1], **Hayala N. Curto**[1]

[1]Instituto de Ciências Exatas e Informática
Ciência da Computação – Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte, MG – Brasil

leon.ferreira@sga.pucminas.br

Resumo. *Este trabalho propõe uma abordagem automatizada para a migração de dados não estruturados a bancos de dados relacionais, utilizando modelos de linguagem de larga escala (LLMs). A metodologia apresenta técnicas para extrair, validar e transformar dados textuais complexos em entidades estruturadas. Foram avaliados cinco modelos (GPT-4o, Qwen-2.5-Max, Dolphin-2.9, Gemini-2.5-Flash e Mistral-3.1-24b) em três conjuntos distintos de dados (QuestionsDB, SoccerDB e SpotifyDB), com variações de formato, volume e complexidade. Os resultados indicam que modelos com maior capacidade contextual alcançam métricas elevadas, mesmo em cenários com milhares de instâncias e múltiplos pedaços de texto processados em paralelo. Ao alinhar a escolha do modelo ao conjunto de dados, a proposta demonstra viabilidade prática para uso em fluxos modernos de dados, especialmente em tarefas como web scraping, ingestão documental e transformação semântica escalável.*

1. Introdução

A migração de dados não estruturados para sistemas de banco de dados modernos tornou-se uma necessidade cada vez mais frequente no cenário tecnológico atual. Com o aumento exponencial na produção de dados provenientes de fontes diversas como textos, documentos, tabelas e multimídia, as empresas enfrentam o desafio de transformar tais dados em informações estruturadas para facilitar a análise e o processamento em sistemas relacionais tradicionais [Schelter et al. 2018]. Conforme discutido por Schelter et al. (2018), abordagens manuais de migração são propensas a erros humanos, comprometendo a integridade e aumentando o tempo necessário para o processamento. Para mitigar esses desafios, diversas ferramentas de migração de dados estruturados têm sido desenvolvidas, como o Talend Open Studio [1] ou o Apache NiFi [2]. Ambas se destacam por sua capacidade de automação de fluxos de dados em tempo real, sendo especialmente úteis em ambientes que requerem alta escalabilidade e flexibilidade. Porém, são ferramentas que necessitam de modelagem e controle humano nas etapas de extração e transformação de informação.

O avanço de tecnologias da IA generativa abre novas oportunidades para automação de tarefas complexas, que discutem aplicações em tradução automática e transformação de dados [Brants et al. 2023]. A aplicação de IA no contexto de migração de dados não estruturados pode automatizar o processo de extração e

[1]https://www.talend.com/products/talend-open-studio/
[2]https://nifi.apache.org/

transformação de dados, garantindo a integridade e a consistência dos mesmos durante a transição [Putten 2024]. O processo de **ETL (Extract, Transform, Load)** [Kimball and Caserta 2004] é fundamental nesse contexto, pois envolve a extração de dados de sistemas de origem, a aplicação de transformações para garantir a qualidade e a conformidade, e o carregamento dos dados em sistemas de destino.

Os modelos de linguagem de grande escala (*LLMs*) [Gao et al. 2020] podem desempenhar um papel crucial na automação do processo de *ETL*. Esses modelos pré-treinados utilizam aprendizado profundo para interpretar textos que representam documentos e gerar representações estruturadas adequadas para bancos de dados relacionais. Modelos como *GPT*, *Gemini*, *Mistral* e *Dolphin*, amplamente utilizados para tarefas de geração de texto e classificação [Putten 2024], podem ser utilizados também para a identificação de entidades, oferecendo um meio-termo entre análise e extração de padrões em textos [Devlin et al. 2019].

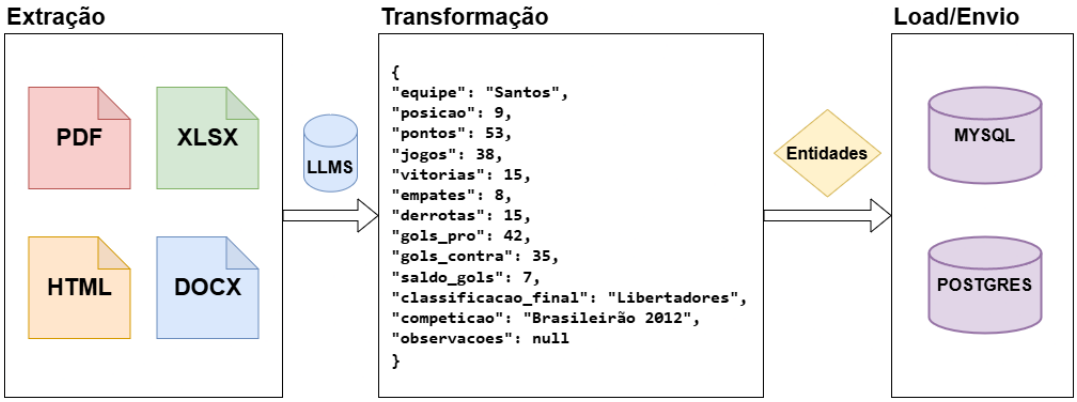

Figura 1. Definição do Fluxo ETL para documentos não estruturados

Este estudo tem como objetivo avaliar a viabilidade da utilização de *LLMs* no contexto da migração de dados não estruturados para formatos estruturados, com ênfase na preservação da integridade dos dados e na capacidade de análise de grandes volumes de informação. Os resultados obtidos em testes práticos permitirão definir a viabilidade do uso de modelos de linguagem para extrair, mapear e validar entidades em processos reais de *ETL* para documentos.

O restante do trabalho está organizado da seguinte forma: inicialmente, realiza-se uma revisão dos trabalhos relacionados. Em seguida, descreve-se a metodologia proposta para a migração de dados, detalhando cada etapa do processo. Depois, são apresentados os testes conduzidos para avaliar a abordagem, detalhando os diferentes conjuntos de dados e os modelos de linguagem empregados, seguidos por uma análise comparativa dos resultados obtidos em cada cenário. Por fim, o artigo encerra-se com as conclusões sobre a viabilidade da metodologia, além de sugestões para trabalhos futuros, indicando melhorias e novas linhas de pesquisa a serem exploradas.

2. Trabalhos relacionados

A evolução dos processos de *ETL* (Extração, Transformação e Carga) tem sido marcada por uma busca contínua por automação para superar os desafios de escala e confiabilidade. Tradicionalmente, esses processos eram manuais e propensos a erros, o que compromete a integridade dos dados e aumenta o tempo de processamento. Nesse contexto, Schelter

et al. [Schelter et al. 2018] apresentaram uma metodologia de verificação de qualidade de dados automatizada para sistemas complexos, que reduz o impacto de erros humanos e melhora a integridade e precisão dos dados processados.

Avançando nessa linha de automação, a aplicação de inteligência artificial (IA) e aprendizado de máquina começou a ganhar força, especialmente no contexto de governança e gerenciamento de dados mestres. Riesener et al. [Riesener et al. 2022], por exemplo, propuseram uma metodologia para gestão automatizada de dados mestres utilizando IA, que facilita o processamento e melhora a qualidade dos dados migrados. Juntos, esses trabalhos estabelecem a importância da automação para garantir a qualidade e a consistência em pipelines de dados, evoluindo de sistemas de verificação para a aplicação direta de IA.

Com o recente avanço da IA generativa, os modelos de linguagem de larga escala (*LLMs*) surgiram como uma tecnologia promissora para preencher essa lacuna. Brants et al. [Brants et al. 2023] destacam que os *LLMs*, já amplamente aplicados em geração de texto e análise de documentos, também permitem maior flexibilidade ao compreender e converter diferentes tipos de mídia, apontando para seu potencial na transformação de dados não estruturados em formatos estruturados presente neste trabalho.

Essa capacidade tem sido explorada em aplicações práticas que visam modernizar a interação com sistemas de dados. Metodologias recentes exploram *LLMs* para democratizar o acesso e a manipulação de dados, como descrito por Korat [Korat 2024], que aponta a capacidade do *LangChain* em transformar consultas de linguagem natural em *SQL*. De forma similar, Chiara Van Der Putten [Putten 2024] investigou o uso de IA generativa em processos clássicos de *ETL*, mas com foco em dados já estruturados, produzindo como artefato final um documento *XML* que espelha o esquema relacional de origem.

Embora essas abordagens demonstrem o valor dos *LLMs* em pipelines de dados, elas operam em cenários onde a estrutura de dados de origem ou de destino já é bem definida. A proposta de Korat [Korat 2024] facilita a consulta a dados estruturados, e o trabalho de Van Der Putten [Putten 2024] otimiza a migração entre sistemas estruturados. A principal contribuição deste trabalho, portanto, é diferenciar-se ao focar na migração de dados fundamentalmente não estruturados (texto livre, tabelas em *PDFs*, etc.) para um modelo relacional fora do fluxo de *ETL* tradicional, utilizando a capacidade semântica dos *LLMs* como o motor central do processo de transformação e estruturação.

3. Metodologia

A abordagem enfrentou desafios relacionados à alta complexidade envolvida na conversão de dados não estruturados de documentos para um modelo relacional com dezenas de atributos, além do elevado custo computacional decorrente do uso de *LLMs* para análise semântica em larga escala. Para mitigar essas limitações, a metodologia proposta adota uma estratégia híbrida, que combina técnicas clássicas de processamento de documentos com otimizações específicas para modelos de linguagem. As etapas desse processo de transformação são representadas na Figura 2.

3.1. Extração inicial dos Dados

A primeira etapa consiste na extração dos dados estruturais do banco de dados relacional alvo. Foram utilizadas técnicas de reflexão de banco de dados [Date 2004], onde são recuperadas informações críticas como nomenclatura de colunas, tipos de dados específicos

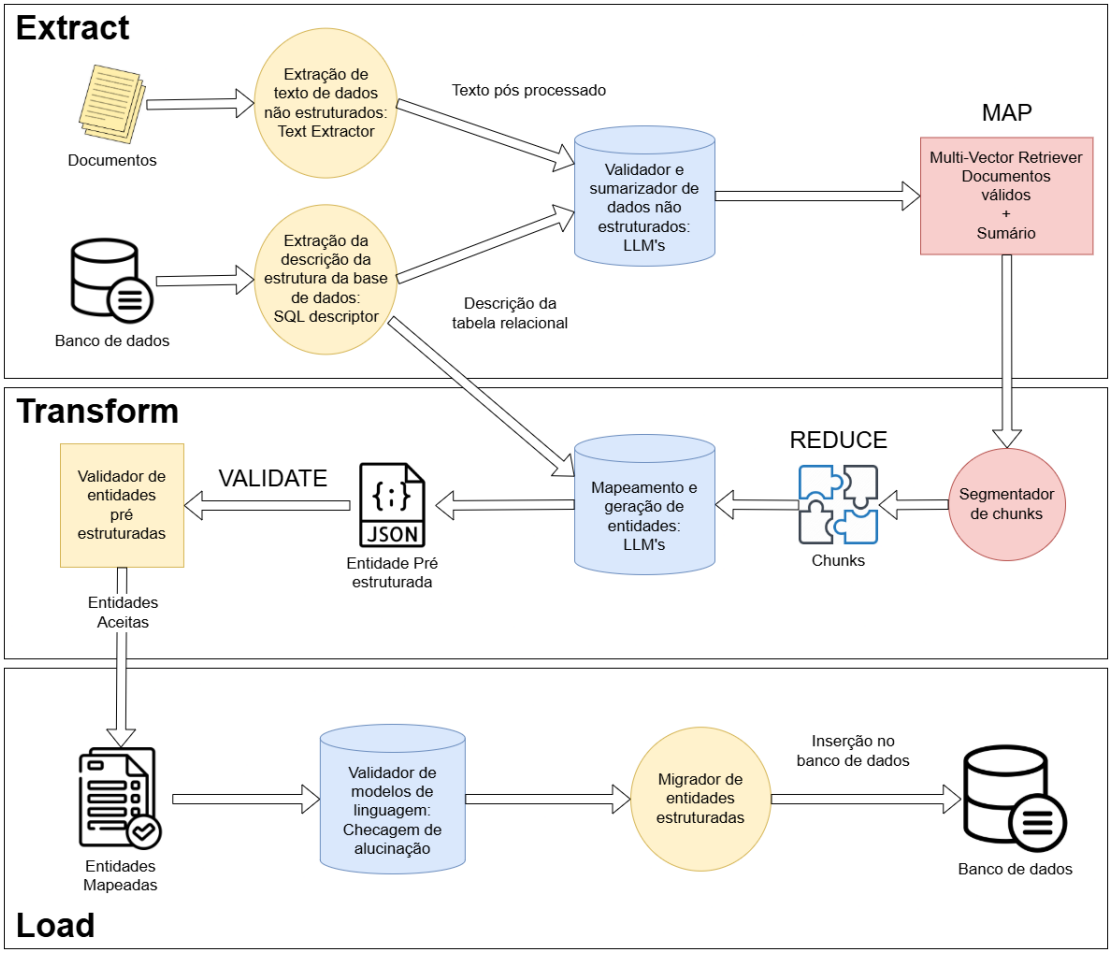

Figura 2. Representação de cada etapa da abordagem

do *SGBD*, restrições de nulidade, valores padrão e mecanismos de auto-incremento. Essa abordagem permitiu capturar não apenas a sintaxe, mas também a semântica das regras de negócios implementadas na estrutura das tabelas.

A representação estruturada em *JSON* gerada nesta fase serve como esquema diretor para todo o processo de transformação. A estrutura formalizada atua como contrato de dados, garantindo que as entidades geradas respeitem a integridade referencial e estrutural do modelo relacional, mesmo quando originárias de fontes não estruturadas.

A segunda etapa do processo concentrou-se na transformação de dados heterogêneos em representação textual unificada. A detecção automática de formatos de documentos e a recuperação de metadados estruturados (autor, datas, idioma) permitiram estratégias de segmentação especializadas como divisão por linhas para documentos paginados, expressões regulares para estruturas *JSON/XML*, e métodos recursivos aplicados a *markdown e HTML*. Cada texto extraído no processo anterior foi submetido a um pós-processamento, responsável por tratar de remover linhas vazias, nulas ou caracteres inválidos.

3.2. Validação e sumarização dos Dados não Estruturados

A etapa seguinte é responsável por identificar se os documentos enviados para o processo fazem parte do conjunto descrito pela base de dados. Para isso, foram rea-

lizadas consultas a modelos generativos de linguagem que atuam como mecanismo de verificação semântica, implementando um paradigma de *Schema-Aware Validation* [Batini and Scannapieco 2016]. Através de *prompts* estruturados com restrições contextuais [Liu et al. 2023], o sistema avalia a relevância documental em relação ao esquema relacional alvo, combinando análise léxica padrão com compreensão contextual. A implementação dessa abordagem utilizou a técnica de processamento paralelo massivo, onde múltiplos processos avaliam documentos simultaneamente.

Os agentes responsáveis por cada instância de validação devem retornar um texto contendo informações básicas juntamente com o sumário do assunto principal dos documentos validados com sucesso. Com esse agregador de contexto adicionado aos metadados, o agente responsável por mapear entidades é guiado a inferir informações que podem não estar presentes no grupo do *chunk* atual, como nomes de objetos, tópico principal do arquivo e até dados intrínsecos ao processo de *ETL*. Essa abordagem é baseada na solução `Multi-Vector Retriever` [3] descrita em um artigo no blog *Langchain*, que pode ser vista na figura 3.

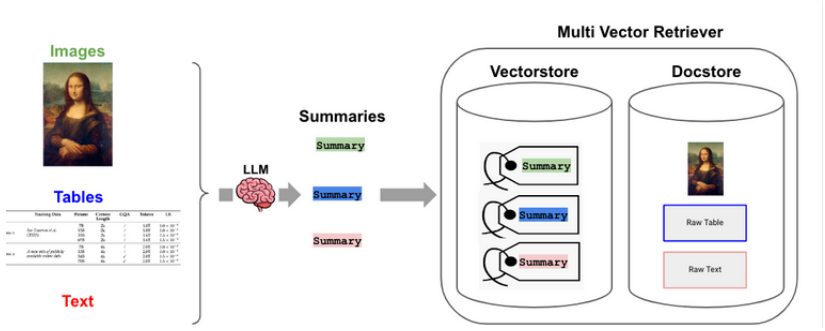

Figura 3. Imagem proposta pelos autores do Multi-Vector Retriever

Ao final do processo anterior, empregaram-se técnicas de *chunking* inteligente [Bialas 2023], particionando o texto bruto extraído em unidades textuais coerentes que preservam relações semânticas. Essa abordagem mitiga problemas de truncamento em modelos de IA [Meng et al. 2024], dividindo o texto em pedaços menores sem remover informações. Nesta etapa do processo, foi fundamental evitar o uso de técnicas de vetorização com indexação de texto, como as propostas por [Bialas 2023], frequentemente aplicadas em sistemas de recuperação aumentada por geração (*RAG*). Esse tipo de abordagem pode resultar em perda irreversível de informações contextuais essenciais, comprometendo irreversivelmente a integridade dos dados e impactando negativamente a acurácia na extração de entidades exatas a partir de conteúdos não estruturados.

3.3. Mapeamento e Geração das Entidades

A etapa central do processo de *ETL* implementa um paradigma de *Schema-Guided Generation* [Liu et al. 2023], onde modelos generativos de linguagem (*LLMs*) transformam fragmentos textuais em entidades estruturadas alinhadas ao esquema relacional. O processo utilizou a descrição da tabela extraída nos passos anteriores como o contrato de dados, orientando a *LLM* na identificação de padrões no texto que revelam possíveis entidades.

[3] `https://blog.langchain.dev/semi-structured-multi-modal-rag/`

Os textos de *prompts* no contexto de *ETL* distribuído foram enviados para o modelo a fim de definir um papel claro, explicitar princípios-chave e estabelecer um formato de resposta rígido, incluindo o tipo de dados e as obrigações semânticas [Liu et al. 2023], conforme pode ser visto na Tabela 1. O processamento da informação foi feito de forma distribuída, com isolamento de contexto por *chunk*, seguindo o modelo *Map-Reduce-Validate* [Dean and Ghemawat 2004], no qual cada agente é responsável por uma pequena porção da tarefa, que foi executada de forma paralela.

Elemento	**Descrição**
Papel do modelo	Processador *ETL* distribuído extraindo dados de *chunks*.
Princípios de controle	Isolamento completo de *chunks*, proibição de suposições com base em partes externas ao *chunk* e rejeição de preenchimento de campos auto-incrementais ou chaves primárias.
Formato de Saída	Retornar `JSON array` válido conforme a descrição da tabela fornecida.
Reforço de regras	**NUNCA** completar IDs, **NUNCA** assumir dados ausentes, **NUNCA** fundir objetos parciais.

Tabela 1. Diretrizes no prompt para o mapeador de entidades

Seja D o conjunto de dados a ser processado, $m = 100$ o tamanho máximo de cada chunk, $f(c_i)$ a resposta em *JSON* retornada pelo *LLM* e R o conjunto de entidades mapeadas finais. Particione D em

$$k \;=\; \left\lceil \tfrac{|D|}{m} \right\rceil \quad \text{pedaços:} \quad \{c_1,\, c_2,\, \ldots,\, c_k\}, \quad |c_i| \le m,\; \forall\, i = 1, \ldots, k.$$

Então,

$$R \;=\; \bigoplus_{i=1}^{k} f(c_i) \;\equiv\; \text{concatenar}\big(f(c_1),\, f(c_2),\, \ldots,\, f(c_k)\big),$$

onde as avaliações $f(c_1),\, \ldots,\, f(c_k)$ ocorrem em paralelo.

Esse método apresentou uma melhoria ao usar paralelismo para processar múltiplos *chunks* simultaneamente, reduzindo significativamente o tempo total de execução, pois as requisições ao agente *LLM* são feitas de forma assíncrona. Essa estratégia, implementada por meio de diversos processos em paralelo, permite que o sistema realize várias tarefas simultâneas sem bloquear outras operações.

3.4. Validação das Entidades Estruturadas

Para a etapa de validação de entidades é importante empregar uma camada de validação sistemática, como base para validar os verdadeiros positivos, foi necessário empregar um arquivo em *JSON* (*testset*) contendo a estrutura esperada de todas as entidades para o documento avaliado (*ground truth*), no qual as saídas de cada *LLM* são comparadas. Foi necessário garantir a integridade referencial das tabelas no banco de dados por meio da rejeição ativa de inserções que violem restrições de chave primária ou regras de preenchimento automático. O conjunto de métricas selecionadas para validar as entidades geradas abrange medidas clássicas de recuperação de informação como *precision*, *recall*, *F1-score*

e *Jaccard Similarity* [Tan et al. 2005], representadas por:

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|}, \qquad \text{Recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|},$$
$$\text{F}_1\text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

onde:

- TP (True Positives): representam acertos e correspondem ao número de atributos corretamente extraídos pelo modelo LLM que também estão presentes no *testset*.
- FP (False Positives): representam extrações incorretas e correspondem ao número de atributos extraídos pelo modelo LLM que não existem ou divergem significativamente dos atributos presentes no *testset*.
- FN (False Negatives): representam omissões na extração e correspondem ao número de atributos que estavam presentes no *testset*, mas não foram extraídos pelo modelo LLM.

A sobreposição entre os conjuntos utiliza os pares "campo=valor" A e B, resultantes das entidades mapeadas pela *LLM* e do *testset*, para definir sua similaridade textual:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Para calcular TP, FP e FN, bem como a interseção na similaridade de Jaccard, utilizou-se correspondência aproximada baseada na distância de Levenshtein [Levenshtein 1966], considerando equivalentes valores de atributos com pelo menos 80% de similaridade.

Paralelamente, foram calculados o *Conformity Rate* para medir a aderência dos tipos de entidade ao esquema relacional e a *Unknown Rate* para identificar ocorrências de elementos não mapeados ou incompatíveis. Violações críticas, como *Missing Mandatory Fields* (ausência de campos obrigatórios na entidade relacional) e *Data Type Errors* (erros de tipagem em campos específicos), foram classificadas como falhas bloqueantes, interrompendo o processo de migração.

3.5. Autoavaliação dos Modelos de Linguagem

Em complemento à avaliação quantitativa, os mesmos modelos que geraram as entidades foram instruídos a realizar um processo de autoavaliação, examinando a coerência, a completude e, principalmente, a fidelidade das respostas em relação ao texto original. A relevância dessa abordagem é corroborada por estudos como os da `OpenAI` [OpenAI 2023], que ressaltam seu papel fundamental na mitigação de alucinações.

O problema da alucinação manifesta-se quando o modelo gera informações factualmente incorretas, inconsistentes ou que não podem ser sustentadas pelo conteúdo de origem. No contexto deste trabalho, uma alucinação poderia, por exemplo, levar um modelo a inferir incorretamente o nome de uma coluna qualquer da entidade, ou preencher o campo com dados não existentes no *chunk* de texto, violando as diretrizes explícitas do *prompt*.

Para a validação semântica, a qualidade da resposta do modelo é medida por meio da análise de seu grau de alucinação, do reconhecimento de suas próprias

limitações (autoconsciência), da abrangência da cobertura contextual e da capacidade de recusar solicitações inadequadas. Estes indicadores, baseados em trabalhos como os de [Hendrycks et al. 2021, Radford et al. 2019], são consolidados em uma métrica de aceitação. Caso o nível de confiabilidade resultante seja inferior a 90%, o processo de migração daquela entidade é imediatamente interrompido.

3.6. Migração dos Dados Estruturados

A etapa de *Load* do processo *ETL* é iniciada após as validações bem-sucedidas das entidades geradas, com foco em inserir cada entidade no banco de dados relacional. Para garantir desempenho e consistência transacional, os registros foram inseridos em lotes (*batch inserts*) configuráveis, de acordo com o parâmetro de *chunk size* definido com o valor de 100 itens por operação. Cada lote foi submetido dentro de uma transação atômica, de modo que falhas em qualquer parte do processo acionem o processo de desfazer os dados, evitando a persistência parcial de dados inconsistentes. Essa consistência transacional é fundamental para o processo geral de *ETL*, pois mantém a integridade dos dados inseridos no banco, assegurando que as transformações anteriores (extração e transformação) sejam refletidas corretamente no sistema de destino.

4. Testes

Os testes foram realizados utilizando 3 grupos distintos de bases de dados [4], cada uma das bases é descrita a seguir:

- **QuestionDB**: Conjunto de 30 questões com respostas para dúvidas de alunos do ensino fundamental. Todas as questões foram adicionadas em tabelas no formato de *XLSX* e tabelas no formato de *DOCX*. Esse é um banco de dados semelhante ao utilizado por Chiara Van Der Putten no seu trabalho de 2024 [Putten 2024]. É o *dataset* mais simples em questão de processamento, contendo 261 KB em arquivos de formato tabular simples;
- **SoccerDB**: Conjunto de documentos no formato de *PDFs* contendo dados sobre o campeonato brasileiro de futebol entre 2014 e 2020. Os documentos foram extraídos da *wikipédia* pública, contendo entre duas a três páginas de conteúdo. Foi inserido um campo chamado de competição, que deve ser inferido dos metadados de sumarização como forma de teste de abrangência para os modelos. Este *dataset* contém 1,80 MB dividido em arquivos *PDFs* complexos, com um processamento moderado de informações textuais sobre o campeonato brasileiro em análise;
- **SpotifyDB**: Conjunto de dados obtido através de consulta aos dados públicos do aplicativo *Spotify*. Playlists públicas salvas em arquivos de texto contendo músicas, cada lista contém músicas de diferentes gêneros em formato de texto plano. Ao todo o conjunto possuí 1500 músicas, um total de 2,74 MB ao todo em arquivos *TXT* e um arquivo resultante esperado de 16.654 linhas de resposta em *JSON*. Envolve processamento massivo de *chunks* por ter um volume considerável de dados em cada arquivo e informações específicas como nomes de autores e músicas salvos em diferentes alfabetos e estruturas.

Os seguintes modelos foram selecionados para testar a abordagem de *ETL*:

- `dolphin-2.9` – Modelo gratuito baseado no `LLaMA`;

[4] Testes e bases de dados: `https://zenodo.org/records/15758992`

- `qwen-2.5-max` – Modelo gratuito treinado pela `Alibaba`;
- `gpt-4o` – Modelo pago treinado pela `OpenAI`;
- `gemini-2.5-flash` – Modelo pago treinado pela `Google`;
- `mistral-small-3.1-24b` – Modelo gratuito treinado pela `Mistral`.

A escolha dos modelos acima busca balancear testes com *LLM's* de acesso livre e modelos pagos e restritos. Para avaliar o desempenho da abordagem, um algoritmo contendo todo o processo de *ETL* foi implementado na linguagem Java 17. Após isso, os experimentos foram conduzidos em um ambiente configurado com as seguintes especificações:

- **Sistema Operacional:** Windows 11 Pro 24H2;
- **Processador:** AMD Ryzen 7 5700X 8 núcleos e 16 Threads;
- **Memória RAM:** 32 GB DDR4;
- **Placa de vídeo:** RTX 4060TI;
- **Memória Secundária:** SSD Kingston KC3000 1TB.

5. Resultados

Para todos os modelos generativos, os testes foram realizados com temperatura de resposta igual a 0,2 e tamanho total de resposta de 12.000 *tokens*, essa definição unitária ajudou a validar os resultados com os mesmos parâmetros. Cada modelo foi submetido a uma sessão individual de processamento, na qual três documentos eram processados em paralelo, cada um subdividido em janelas de 16 *chunks*, também executadas em paralelo.

5.1. QuestionsDB

No *dataset* **QuestionsDB**, os modelos **qwen-2.5-max**, **dolphin-2.9** e **gpt-4o** atingiram as maiores métricas, com pontuação máxima de 1,0 em todas, evidenciando a qualidade da extração e conversão estruturada das entidades em cenários simples.

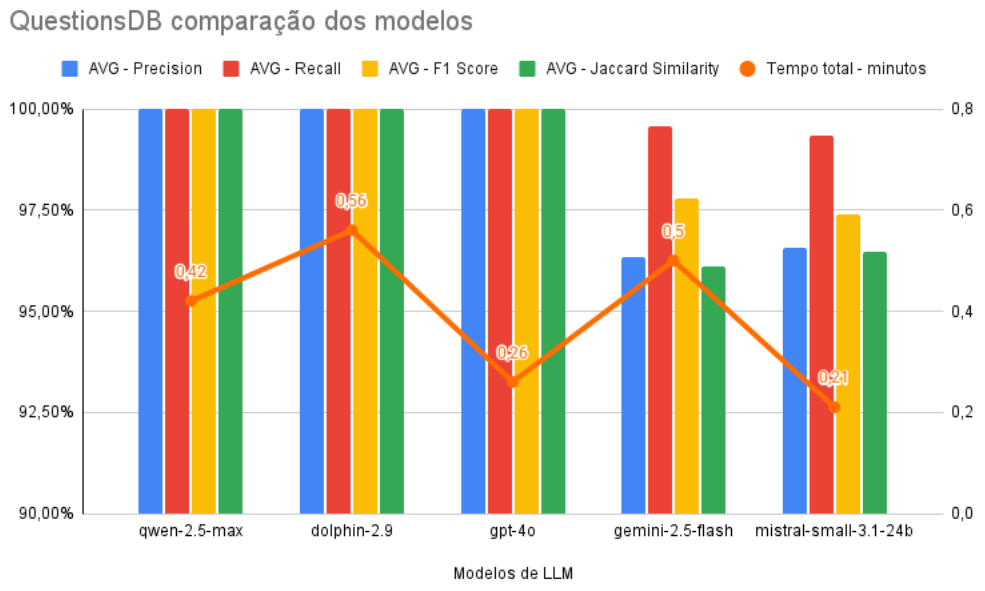

Figura 4. Métricas dos modelos para o dataset QuestionDB

Os modelos **gemini-2.5-flash** e **mistral-small-3.1-24b** apresentaram métricas com pequenas variações no desempenho, respectivamente com precisão de 0,963 e 0,965, recall de 0,995 e 0,993, F1-Score de 0,977 e 0,974, e similaridade estrutural de 0,96 e 0,964. Apesar de levemente inferiores, seus resultados permanecem acima de 0,96, confirmando sua confiabilidade para esse cenário. O tempo de execução variou entre 26s (**mistral-small-3.1-24b**) e 56s (**dolphin-2.9**), sem impacto nos resultados das entidades.

Para este conjunto de dados, a abordagem demonstrou-se eficiente para todos os modelos. Por se tratar de um *dataset* relativamente simples e com uma estrutura tabular que favorece a extração, todos apresentaram bom desempenho com qualidade alta.

5.2. SoccerDB

No *dataset* **SoccerDB**, que envolve extração de texto não estruturado em PDF, observou-se uma considerável variação de desempenho entre os modelos (Figura 5). O **gpt-4o** destacou-se como o mais eficaz, alcançando precisão de 0,944, recall de 0,936 e F1-Score de 0,940. Em seguida, **qwen-2.5-max** obteve F1-Score de 0,892 e **dolphin-2.9** registrou F1-Score de 0,851, demonstrando desempenho robusto, embora inferior ao líder. O **gemini-2.5-flash** apresentou métricas intermediárias (precisão de 0,823, recall de 0,769, similaridade estrutural de 0,626), indicando queda na similaridade estrutural, enquanto o **mistral-small-3.1-24b** obteve os índices mais baixos (F1-Score de 0,588, precisão de 0,499, similaridade estrutural de 0,430), evidenciando dificuldade na extração consistente e na ordenação dos campos.

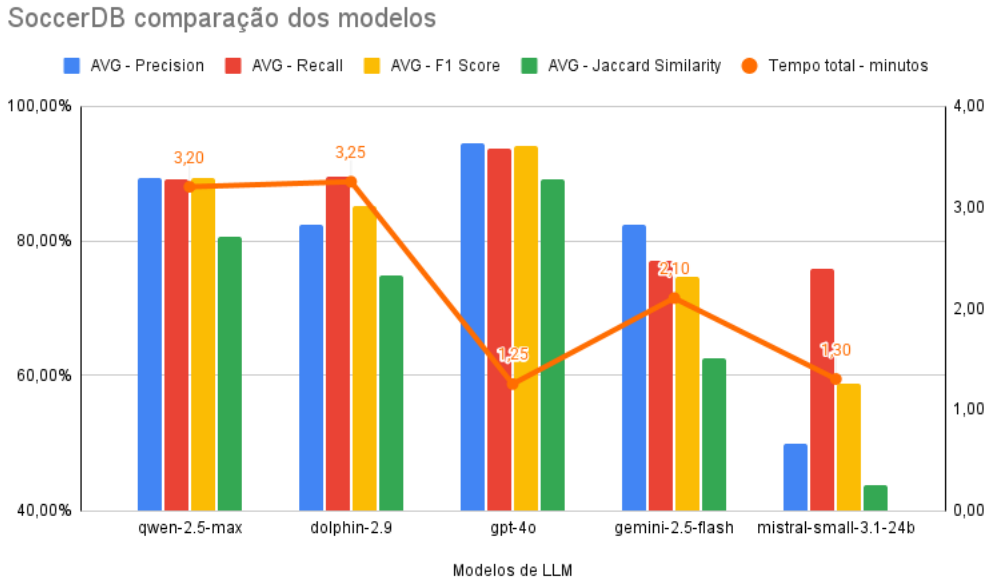

Figura 5. Métricas dos modelos para o dataset SoccerDB

Em termos de tempo de execução, o **gpt-4o** combinou as melhores métricas com o menor tempo de resposta (1:25). Embora o **mistral-small-3.1-24b** tenha apresentado tempo semelhante (1:30), seu desempenho foi significativamente inferior. Modelos mais lentos, como **dolphin-2.9** (3:25) e **qwen-2.5-max** (3:02), não superaram o **gpt-4o**, indicando que maior tempo de execução não se traduz necessariamente em melhores resultados em tarefas *ETL* complexas.

A sumarização dos documentos inserida como metadado de contexto permitiu tratar de forma mais eficaz a inferência de campos críticos, como a coluna `competição`, reduzindo o efeito de desconhecimento e melhorando a extração de informações estruturadas em modelos com grande capacidade de inferência de informação.

5.3. SpotifyDB

O *dataset* final **SpotifyDB**, com alto volume de dados por arquivo, desafiou os modelos a manterem consistência em múltiplos *chunks*. A Figura 6 mostra que os modelos **gpt-4o** e **qwen-2.5-max** apresentaram um desempenho comparável entre ambos, com F1-Scores de 0,990 e 0,988, respectivamente, além de altas similaridades estruturais (0,981 e 0,979). O tempo de execução difere muito entre os dois modelos, (2:17) e (4:32), indicando que o tempo de execução e as altas qualidades de resposta não estão correlacionados.

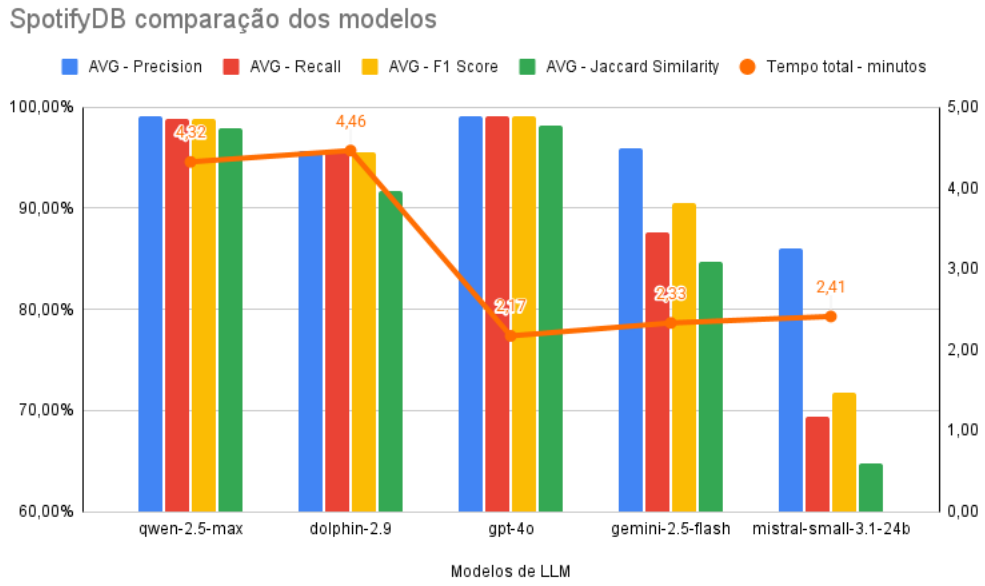

Figura 6. Métricas dos modelos para o dataset SpotifyDB

O modelo **dolphin-2.9** também obteve métricas altas (F1-Score de 0,955 e similaridade de 0,917), mostrando-se eficaz mesmo com o maior tempo de execução (4:46 min). O **gemini-2.5-flash**, apesar de um bom F1-Score (0,904), demonstrou menor consistência na estrutura (0,847), refletindo dificuldades com a coerência dos dados em larga escala. O **mistral-small-3.1-24b** teve a menor performance geral, com F1-Score de 0,717 e similaridade estrutural de apenas 0,646, indicando limitações na manipulação de dados extensos e paralelos.

A segmentação desse *dataset* em pequenos pedaços de texto testou ao máximo o processamento paralelo para todos os modelos e foi possível notar que modelos com maior capacidade contextual lidam melhor com mais *chunks* de uma vez, garantindo métricas e resultados melhores para o maior *dataset* utilizado para validar o desempenho.

5.4. Análise Geral dos Resultados

Os resultados globais na figura 7 indicam uma correlação direta entre a capacidade contextual dos modelos e seu desempenho em extrações de complexidade variada. É fundamental observar que todos os modelos processaram as entidades sem incorrer em erros

bloqueantes, como *Missing Mandatory Fields* ou *Data Type Errors*, e sem falhas em validações semânticas de alucinação.

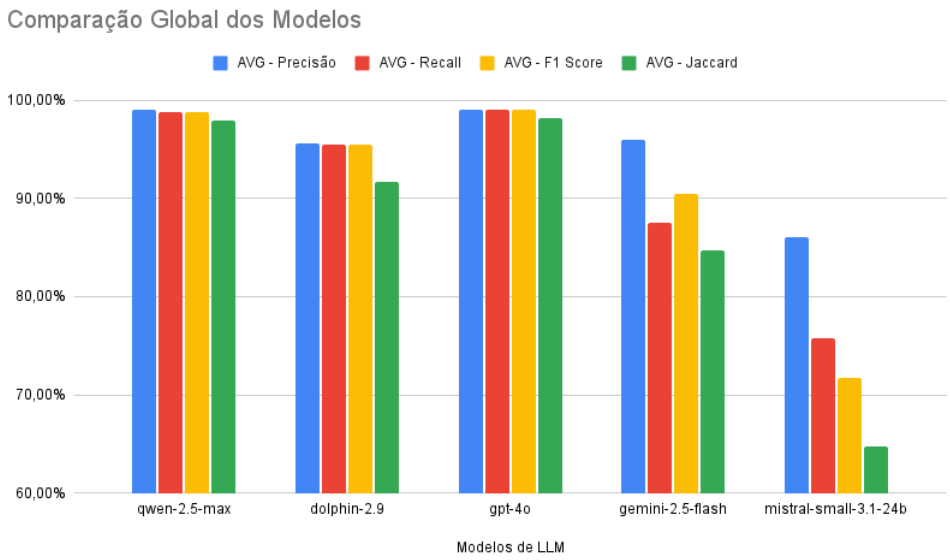

Figura 7. Comparação das métricas finais entre todos os modelos

Para verificar a existência de diferenças estatisticamente significativas entre os modelos nas métricas de precisão, recall, F1-Score e Jaccard, aplicou-se o teste de Friedman [Friedman 1937], seguido do procedimento post-hoc de Nemenyi [Nemenyi 1963]. Em todos os casos, o teste de Friedman indicou valor de $p < 0,01$, comprovando que nem todos os modelos apresentam desempenho equivalente. Os modelos ocuparam, respectivamente, as primeiras colocações de acordo com o ranking médio: **gpt-4o**, **qwen-2.5-max**, **dolphin-2.9**, **gemini-2.5-flash** e **mistral-small-3.1-24b**.

Além disso, o post-hoc de Nemenyi demonstrou que **gpt-4o** superou de maneira significativa **qwen-2.5-max** nas métricas F1-Score ($p = 0,02$) e Jaccard ($p = 0,01$), não sendo observadas diferenças significativas em precisão e recall. **Dolphin-2.9** destacou-se em terceiro lugar, apresentando desempenho estatisticamente diferenciado em relação aos dois primeiros ($p < 0,05$), mas mantendo-se à frente dos demais modelos, os quais revelaram queda acentuada em recall, sugerindo maior propensão à omissão de dados. Este procedimento estatístico reforça que, embora vários modelos atinjam médias acima de 95%, apenas **gpt-4o** e **qwen-2.5-max** mantêm desempenho consistentemente superior, justificando sua adoção em cenários de extração crítica de entidades.

6. Conclusão

Neste trabalho, avaliou-se uma abordagem automatizada de *ETL* para a migração de dados não estruturados para bancos de dados relacionais, utilizando cinco modelos de linguagem de larga escala (*LLMs*). A análise dos resultados, validada estatisticamente, demonstrou que a escolha do modelo é um fator crítico para o sucesso da migração. O teste de Friedman confirmou que existem diferenças de desempenho estatisticamente significativas entre os modelos avaliados ($p < 0,01$), com o *ranking* médio posicionando **gpt-4o**, **qwen-2.5-max** e **dolphin-2.9** nas primeiras colocações. O teste *post-hoc* de Nemenyi reforçou

a superioridade do **gpt-4o**, que não só apresentou o menor tempo de execução , mas também superou significativamente o segundo colocado, **qwen-2.5-max**, em métricas-chave de desempenho estrutural como F1-Score e Jaccard Similarity.

A análise expõe uma importante troca entre o custo e o desempenho. Embora o modelo proprietário **gpt-4o** seja estatisticamente superior, modelos de acesso livre como o **qwen-2.5-max** e o **dolphin-2.9** surgem como alternativas altamente competitivas, entregando resultados robustos e com alta fidelidade em cenários complexos. Por outro lado, os modelos **gemini-2.5-flash** e **mistral-small-3.1-24b** apresentaram um desempenho estatisticamente inferior, com uma queda acentuada no *recall*, indicando maior propensão à omissão de dados e tornando-os menos adequados para ambientes de produção com requisitos de alta precisão.

Um ponto fundamental é que nenhum dos modelos incorreu em erros bloqueantes (como *Missing Mandatory Fields* ou *Data Type Errors*) ou falhou nas validações de alucinação, o que reforça a integridade estrutural e a confiabilidade do processo. Isso abre portas para a implementação em *pipelines* de dados reais para automatizar tarefas como a digitalização e extração de conteúdo de documentos, a ingestão de informações via *web scraping* e a transformação de dados legados em entidades.

Como trabalhos futuros, destaca-se a possibilidade de testar novas estratégias de adaptação de *prompts* para otimizar o desempenho de cada modelo individualmente. Além disso, a integração com metadados adicionais e validação semântica contínua pode melhorar a confiabilidade e o tempo de processamento. Outra linha promissora é a exploração de arquiteturas especializadas para diminuir o custo monetário de modelos pagos, sem comprometer a precisão.

Em suma, a metodologia proposta é promissora e escalável, contanto que a escolha do *LLM* seja alinhada às especificações do *dataset*. Para modelos com alta capacidade contextual, a abordagem se mostrou eficaz para automatizar a extração a partir de fontes diversas, reduzindo o esforço manual e garantindo a integridade dos dados, reforçando seu potencial para converter dados não estruturados em entidades estruturadas para bancos de dados relacionais.

Referências

Batini, C. and Scannapieco, M. (2016). *Data Quality: Concepts, Methodologies and Techniques*. Springer, Cham, Switzerland.

Bialas, A. (2023). Effective chunking strategies for rag. `https://docs.cohere.com/v2/page/chunking-strategies`. Accessed: 2025-05-05.

Brants, T., Popat, A., Xu, P., Och, F., and Dean, J. (2023). Large language models in machine translation. In *Artificial Intelligence and Neural Networks*, pages 623–630. Springer.

Date, C. J. (2004). *An Introduction to Database Systems*. Pearson/Addison Wesley, Boston.

Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI '04)*, pages 137–150, Berkeley, CA, USA. USENIX Association.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Confe-*

rence of the North American Chapter of the Association for Computational Linguistics, pages 4171–4186.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

Gao, T., Levine, S., and Darrell, T. (2020). Making pre-trained language models better few-shot learners. *Proceedings of the 37th International Conference on Machine Learning*, 119:2027–2036.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Kimball, R. and Caserta, J. (2004). *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, Indianapolis, IN.

Korat, A. S. (2024). Ai-augmented langchain: Facilitating natural language sql queries for non-technical users. *Journal of Artificial Intelligence & Cloud Computing*, 3(3):3–5.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):195:1–195:35.

Meng, Y. et al. (2024). Ranked list truncation for large language model-based re-ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Nemenyi, P. B. (1963). *Distribution-free multiple comparisons*. PhD thesis, Princeton University, Princeton, NJ.

OpenAI (2023). Gpt evaluation: Methodologies for robust assessment. Technical report.

Putten, C. V. D. (2024). Transforming data flow: Generative ai in etl pipeline automatization. Master's thesis, Politecnico di Torino, Corso di laurea magistrale in Data Science And Engineering. In collaboration with Mediamente Consulting Srl.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Technical report, OpenAI.

Riesener, M., Kuhn, M., Lender, B., and Schuh, G. (2022). Methodology for automated master data management using artificial intelligence. In *2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1276–1280.

Schelter, S., Liu, S., Mankowski, D., Markl, V., Chard, K., and Foster, I. (2018). Automating large-scale data quality verification. *The VLDB Journal*, 27(1):15–37.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.