Recuperação de documentos para enriquecimento de dados tabulares

Fernando R. Delbone[1], **Eduardo H. M. Pena**[1]

[1]Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
Campo Mourão – PR – Brasil

Resumo. *Técnicas de descoberta de dados focam principalmente na descoberta de dados por meio de palavras-chave, busca por similaridade entre tabelas ou consultas estruturadas. Tais abordagens pouco exploram a conexão entre tabelas e documentos de texto, que pode viabilizar o enriquecimento de dados tabulares a partir de fontes textuais. Neste trabalho, propomos o problema table-to-documents: dada uma tabela como entrada e uma coleção de documentos em um data lake, recuperar os top-k documentos mais relevantes para essa tabela. Nossas contribuições incluem: uma base de dados adaptada à esse problema; uma solução inicial que explora diversas representações tabela-documento; e a avaliação experimental dessa solução.*

Abstract. *Data discovery techniques focus primarily on discovering data through keywords, similarity search between tables, or structured queries. Such approaches largely overlook the connection between tables and text documents, which can enable the enrichment of tabular data from textual sources. In this work, we propose the table-to-documents problem: given a table as input and a collection of documents in a data lake, retrieve the top-k most relevant documents for that table. Our contributions include: a benchmark dataset adapted to this problem; an initial solution that explores diverse table-document representations; and an experimental evaluation of that solution.*

1. Introdução

A descoberta de dados em grandes repositórios heterogêneos tornou-se um problema central em *dataset search*. Em *data lakes*, portais de dados abertos e repositórios corporativos, usuários precisam localizar tabelas relevantes entre milhares de artefatos com esquemas, conteúdos e granularidades distintas. Grande parte da literatura aborda os problemas de busca em *data lakes* partir do próprio conteúdo tabular, explorando nomes de atributos, valores de células, metadados e relações entre tabelas para apoiar busca por palavras-chave, descoberta de tabelas relacionadas e outras formas de recuperação em *data lakes* [Paton et al. 2023]. Em muitos cenários, no entanto, informações essenciais para interpretar uma tabela não estão nos dados, mas em documentos textuais associados ao seu contexto de uso, como regulamentações, manuais, e relatórios.

Documentos podem complementar a tabela de diferentes maneiras. Eles podem explicar o significado dos atributos, explicitar restrições de uso, registrar definições institucionais e fornecer contexto útil para tarefas posteriores, como verificação de fatos, extração de regras e *table augmentation* [Wu et al. 2014, Freitag et al. 2022]. O problema é que a relação entre tabelas e documentos raramente está disponível de forma explícita.

Este trabalho aborda o seguinte cenário. Dada uma tabela de entrada e uma coleção de documentos textuais, buscamos recuperar os top-k documentos mais relevantes para essa tabela. Chamamos essa tarefa de *table-to-documents*. Ela pode ser entendida como um problema de *dataset search* centrado na recuperação de contexto documental para dados tabulares, em vez da recuperação de outras tabelas. Essa etapa pode servir como base para aplicações de contextualização, validação e enriquecimento de dados.

Como contribuições iniciais, apresentamos a formulação do problema, discutimos sua relação com linhas de pesquisa próximas, propomos e disponibilizamos três conjuntos de dados para avaliação e conduzimos experimentos preliminares com um método adaptado a esse cenário.

2. Definição do Problema

Consideramos um *data lake* composto por um conjunto de tabelas $\mathcal{T} = \{T_1, \ldots, T_n\}$ e um conjunto de documentos textuais $\mathcal{D} = \{D_1, \ldots, D_m\}$. Uma tabela $T \in \mathcal{T}$ é definida por seu esquema $sch(T) = \langle A_1, \ldots, A_p \rangle$ e por seu conjunto de tuplas $tup(T) = \{t_1, \ldots, t_q\}$, em que cada tupla t_ℓ atribui valores aos atributos do esquema. Um documento textual $D \in \mathcal{D}$ é um objeto textual não estruturado, como uma regulamentação, manual, relatório ou descrição.

Dada uma tabela de entrada $T \in \mathcal{T}$, o problema consiste em recuperar os top-k documentos textuais em $\mathcal{D}$ mais relevantes para T, isto é, os documentos que melhor explicam, contextualizam ou enriquecem o conteúdo da tabela. Em outras palavras, desejamos definir, para cada tabela $T_i \in \mathcal{T}$, um ranqueamento sobre $\mathcal{D}$ e retornar os k documentos mais bem posicionados nesse ranqueamento.

3. Método de Recuperação Tabela–Documento

Esta seção descreve nossa abordagem para o problema introduzido na Seção 2. Nosso pipeline compreende três etapas: (i) construção de representações textuais para tabelas e documentos, (ii) codificação dessas representações em um espaço vetorial comum e (iii) recuperação sobre um índice pré-computado.

3.1. Abordagem

Nossa abordagem projeta tabelas e documentos textuais em um espaço vetorial comum, de modo que a relevância entre esses objetos possa ser estimada por similaridade vetorial.

Dada uma tabela $T \in \mathcal{T}$ e um documento $D \in \mathcal{D}$, cada objeto é inicialmente convertido em uma representação textual. Para a tabela, a representação $r(T)$ é construída a partir da serialização de seu esquema $sch(T)$ e de uma amostra de suas tuplas $tup(T)$. Para o documento, a representação $r(D)$ corresponde ao seu conteúdo textual, podendo considerar o texto completo ou apenas partes selecionadas, como título, resumo ou trechos introdutórios. Em seguida, uma função de codificação $e(\cdot)$, mais detalhes abaixo, mapeia essas representações textuais em vetores no mesmo espaço: $\mathbf{v}_T = e(r(T)) \qquad \text{e} \qquad \mathbf{v}_D = e(r(D))$.

A relevância entre T e D é então estimada pela similaridade do cosseno entre seus vetores: $s(T, D) = \frac{\mathbf{v}_T \cdot \mathbf{v}_D}{\|\mathbf{v}_T\| \, \|\mathbf{v}_D\|}$. Com base nessa função de similaridade, associamos a cada tabela $T \in \mathcal{T}$ um ranqueamento dos documentos em $\mathcal{D}$ e retornamos os top-k documentos com maiores valores de $s(T, D)$.

3.2. Projeção de Representações

A eficácia da recuperação depende da forma como tabelas e documentos são convertidos para texto. Para documentos, essa conversão é direta, pois o conteúdo já se encontra em linguagem natural. Para tabelas, contudo, é necessário linearizar uma estrutura bidimensional definida por $sch(T)$ e $tup(T)$ em uma sequência textual compatível com o codificador. Essa escolha afeta quais aspectos da tabela são preservados na representação final, como regularidades intra-coluna, relações entre valores de um mesmo registro, delimitações estruturais e sinais semânticos presentes nos atributos.

Neste trabalho, investigamos quatro famílias para $r(T)$: representações orientadas a linhas, representações orientadas a colunas, representações com marcação estrutural explícita e representações semânticas enriquecidas. Na implementação, essas famílias são instanciadas por nove variantes, descritas a seguir.

Linearização por linha concatena os valores de cada tupla na ordem dos atributos, preservando a coocorrência local entre valores de um mesmo registro [Yin et al. 2020]. *Linearização por coluna* agrupa os valores por atributo antes da transição para a próxima coluna, favorecendo a exposição de regularidades intra-coluna [Suhara et al. 2022]. *Representação por template* converte cada linha em uma sentença em linguagem natural por meio de um molde textual predefinido [Zhang et al. 2023]. *Serialização plana* converte a tabela em uma sequência única com marcadores explícitos de cabeçalho, colunas e linhas [Devlin et al. 2019]. *Markdown* textualiza a tabela em formato Markdown, preservando parcialmente sua organização tabular de forma legível [Sadia et al. 2025]. *Estrutura HTML* representa a tabela com marcações análogas a HTML, explicitando cabeçalhos, células e linhas [Bhagavatula et al. 2013]. *Triplas atributo-valor* reorganiza o conteúdo em triplas que associam posição de coluna, nome do atributo e valor, tornando relações locais mais explícitas [Hulsebos 2024]. *Descrição de esquema* enfatiza metadados da tabela, como número de colunas, tipos, cardinalidade, valores ausentes e exemplos de valores [Hulsebos 2024]. Por fim, *representação contextualizada* incorpora informações globais e estatísticas por coluna, enriquecendo a tabela com sinais agregados além dos valores brutos [Cong et al. 2023].

3.3. Recuperação

Adotamos o SPLADE (*Sparse Lexical and Expansion*) [Formal et al. 2021] como função de codificação $e(\cdot)$ para representar tabelas e documentos por meio de vetores esparsos. Essa escolha busca acomodar o seguinte. Primeiro, a correspondência entre tabela e documento frequentemente depende de âncoras lexicais explícitas, como nomes de atributos, categorias e valores característicos. Segundo, o documento relevante nem sempre repete exatamente os mesmos termos da tabela, o que exige algum grau de expansão semântica. Utilizamos o modelo pré-treinado `naver/splade-cocondenser-ensembledistil`, disponibilizado pela biblioteca *Sentence Transformers*.

A codificação é assimétrica pois a representação textual da tabela $r(T_i)$ é tratada como consulta, enquanto cada documento $r(D_j)$ é tratado como documento indexável. Essa escolha reflete o cenário onde uma tabela de entrada é utilizada para buscar documentos relevantes em um corpus textual fixo.

3.4. Recuperação e Ranqueamento

Os vetores das representações dos documentos em $\mathcal{D}$ são pré-computados e armazenados em cache. Essa decisão reduz o custo recorrente da busca, pois, para cada tabela T_i, apenas sua representação textual precisa ser codificada em tempo de consulta. Em seguida, calculamos a pontuação $s(T_i, D_j)$ para cada $D_j \in \mathcal{D}$, ordenamos os documentos pela pontuação obtida e retornamos os top-k resultados. Essa etapa de busca pode ser implementada com qualquer índice adequado para recuperação vetorial, incluindo estruturas de busca aproximada. Nesta versão preliminar, adotamos busca exaustiva sobre os documentos candidatos. Essa escolha evita perdas introduzidas por aproximações no índice e permite avaliar diretamente a qualidade das representações e da função de pontuação.

4. Construção do Conjunto Tabela–Documento

Como o problema estudado neste trabalho não possui um conjunto de dados pronto para avaliação, construímos uma coleção tabela–documento. A coleção combina documentos sintéticos, gerados a partir de *benchmarks*, e documentos reais, extraídos da Wikipedia. A base final contém 102 tabelas e 1000 documentos textuais. A unidade básica da coleção é o par tabela–documento. Para cada tabela, definimos documentos relevantes, isto é, textos que contêm informações associadas ao conteúdo da tabela, e documentos não relevantes, usados como candidatos negativos na tarefa.

As duas primeiras fontes, Spider e ClaimDB, são adaptações de *benchmarks* consolidados que já relacionam textos curtos a tabelas. Como esses textos não correspondem a documentos completos, usamos LLMs para gerar *pseudodocumentos*: textos sintéticos que simulam documentos descritivos, relatórios ou páginas explicativas que poderiam acompanhar uma tabela em um *data lake*. Para cada tabela, primeiro construímos uma descrição contendo domínio, atributos, tipos de valores e exemplos de conteúdo. Em seguida, uma LLM usa essa descrição e o texto original associado à tabela para gerar um documento narrativo. Nesta etapa, utilizamos o modelo Gemini 3.0-Flash.

Do Spider [Yu et al. 2018], um *benchmark* de *text-to-SQL*, usamos as consultas SQL para identificar as tabelas mencionadas em cada pergunta em linguagem natural. Cada par pergunta–tabela foi usado como base para gerar documentos sintéticos relevantes. Do ClaimDB [Theologitis et al. 2026], um *benchmark* de verificação de fatos com base em evidências tabulares, usamos as afirmações suportadas e contraditórias, pois ambas exigem a consulta a uma tabela de evidência para serem verificadas. Cada par afirmação–tabela foi então usado como base para gerar documentos sintéticos relevantes. Para cada tabela derivada do Spider e do ClaimDB, geramos cinco documentos sintéticos relevantes e cinco documentos distratores. Os documentos relevantes incorporam informações verificáveis a partir da tabela. Os distratores tratam do mesmo domínio temático, mas não incluem evidências extraídas dela. Com isso, a tarefa de recuperação exige distinguir textos apenas relacionados ao tema daqueles que de fato contêm informações associadas ao conteúdo tabular.

A terceira fonte utiliza dados reais extraídos da Wikipedia. Para construí-la, realizamos *scraping* de páginas que continham tabelas, usando termos de busca variados para cobrir diferentes domínios. Para cada página coletada, extraímos o texto do artigo e as tabelas presentes nela, formando pares tabela–artigo. Em seguida, filtramos os pares para manter apenas aqueles com alta sobreposição entre o conteúdo textual do artigo e os

dados da tabela. Essa sobreposição foi estimada com TF-IDF, a partir dos tokens extraídos da tabela e do corpo do artigo. Mantivemos apenas os pares em que a sobreposição atingia pelo menos 10% segundo esse critério.

5. Resultados Preliminares

Os artefatos experimentais, incluindo código (desenvolvido em Python) e materiais de apoio, estão disponíveis em[1]. Nossos experimentos rodaram em uma máquina com processador Intel Core i5, 16,GB de RAM e sistema operacional Ubuntu 22.04.

A avaliação da nossa abordagem é conduzida por meio da métrica *Precision@k* [Manning et al. 2008]: Precision@$k = \frac{\left|D_j \in \text{top-}k | D_j \in \mathcal{R}(T_i)\right|}{k}$, em que $\mathcal{R}(T_i)$ denota o conjunto de documentos relevantes para a tabela T_i, de acordo com o *ground truth*. Em nosso *ground truth*, adotamos $k = 5$ para Spider e ClaimDB, e $k = 1$ para o Wikipedia. A Figura 1 apresenta os resultados da nossa solução nos três benchmarks avaliados.

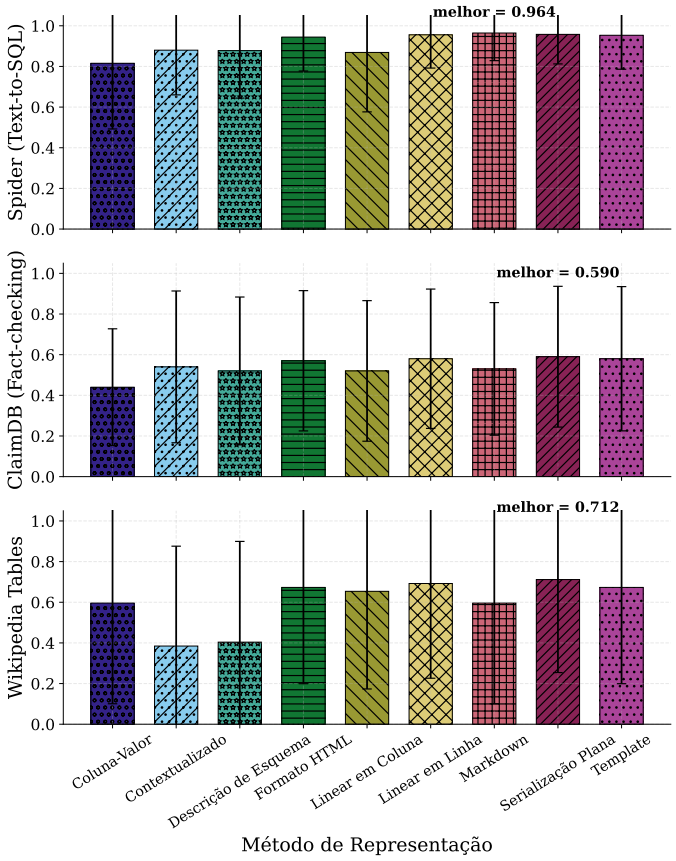

Figura 1. Comparação dos métodos de representação nos datasets Spider, ClaimDB e Wikipedia, com base no Precision@k médio. As barras de erro indicam a variabilidade dentro de cada método.

No Spider, a solução apresenta desempenho alto e relativamente estável entre os métodos de representação. Todas as representações obtêm médias de Precision@k acima de 0,80. O melhor resultado é obtido com Markdown (0,964), seguido de perto por Serialização Plana (0,958), Linear em Linha (0,956) e Template (0,953). A taxa de acerto

[1] https://github.com/FerDelbo/table-to-documents

perfeito também é alta: Markdown, Linear em Linha e Template atingem cerca de 92% de consultas com Precision@k = 1. Esse resultado pode estar relacionado à forma como os documentos foram gerados nesse benchmark. Os documentos relevantes são produzidos a partir da combinação entre a tabela e a pergunta associada a ela. Assim, a pergunta fornece um contexto semântico adicional que orienta a LLM a gerar um texto bastante específico. Já os documentos adversários são gerados apenas com base no nome da tabela, o que produz textos menos específicos e com menor densidade factual. Com isso, a tarefa de recuperação se torna mais simples, pois o modelo pode identificar o documento correto com base em sinais semânticos relativamente superficiais.

O cenário muda nos dados do ClaimDB e da Wikipedia, onde há queda brusca nas médias e aumento da dispersão. No ClaimDB e na Wikipedia, a melhor média é obtida pela Serialização Plana, com valores de 0,590 e 0,712, respectivamente. Além da redução nas médias, os resultados se tornam mais instáveis, com aumento dos desvios-padrão. Essa mudança no ranking entre os *datasets* está relacionada às características de cada conjunto. No ClaimDB, há maior ambiguidade semântica entre esquemas e termos muito próximos. Além disso, os documentos adversários são gerados com base em fatos contraditórios, o que exige que o modelo diferencie os documentos em um nível mais ligado aos dados do que apenas ao tema geral. Ao analisar os documentos recuperados, observamos que, independentemente da técnica de representação utilizada, o modelo frequentemente recupera os documentos adversários. Esse comportamento evidencia uma limitação das representações atuais. Embora consigam capturar a estrutura da tabela, elas ainda falham em explorar adequadamente os valores dos dados. Soma-se a isso uma característica do próprio ClaimDB, que associa os fatos ao esquema, e não a uma tabela específica.

Na Wikipedia, esse efeito é ainda mais forte. Nesse cenário, representações mais enxutas e lexicalmente diretas, como a Serialização Plana, tendem a preservar melhor os sinais úteis para a correspondência entre tabela e documento. Já representações com maior marcação estrutural, como Markdown, funcionam bem em cenários mais controlados, mas tornam-se mais suscetíveis à diluição do sinal semântico quando há maior heterogeneidade nos documentos. Esses resultados estão alinhados à literatura. Trabalhos anteriores mostram que a representação da tabela afeta diretamente a recuperação, seja pela importância dos sinais semânticos e contextuais no ranqueamento [Cong et al. 2023], seja pela perda de informação relevante quando ela não aparece de forma saliente no contexto [Liu et al. 2024].

6. Conclusão

Neste trabalho, formalizamos a tarefa de recuperação de documentos a partir de tabelas (*table-to-documents*). Como contribuição, desenvolvemos uma base de dados, projetada para viabilizar a avaliação e a validação da abordagem proposta. Os resultados experimentais confirmam que o uso de diferentes representações de tabelas para a recuperação de documentos é viável, embora a complexidade da tarefa aponte para novos desafios.

Agradecimentos

Este estudo foi realizado com apoio da Fundação Araucária de Apoio ao Desenvolvimento Científico e Tecnológico do Estado do Paraná e do Instituto Serrapilheira, concedido por meio do Programa de Apoio a Jovens Cientistas (Protocolo n° SER2023371000001 / PI 28/2023).

Referências

Bhagavatula, C. S., Noraset, T., and Downey, D. (2013). Methods for exploring and mining tables on wikipedia. In *IDEA '13*, pages 18–26.

Cong, T., Hulsebos, M., Sun, Z., Groth, P., and Jagadish, H. V. (2023). Observatory: Characterizing embeddings of relational tables. *PVLDB*, 17(4):849–862.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*, pages 4171–4186.

Formal, T., Piwowarski, B., and Clinchant, S. (2021). Splade: Sparse lexical and expansion model for first stage ranking. In *SIGIR '21*, pages 2288–2292.

Freitag, D., Cadigan, J., Sasseen, R., and Kalmar, P. (2022). Valet: Rule-based information extraction for rapid deployment. In *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, pages 524–533.

Hulsebos, M. (2024). *Table Representation Learning*. Phd thesis, Universiteit van Amsterdam. Defended 23 February 2024.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.

Paton, N. W., Chen, J., and Wu, Z. (2023). Dataset discovery and exploration: A survey. *ACM Comput. Surv.*, 56(4).

Sadia, M., Yang, Z., Xiao, Y., Chen, A., and Roy Chowdhury, A. (2025). SQUiD: Synthesizing relational databases from unstructured text. In *EMNLP 2025*, pages 31987–32012.

Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, c., Chen, C., and Tan, W.-C. (2022). Annotating columns with pre-trained language models. In *SIGMOD 2022*, pages 1493–1503.

Theologitis, M., Dammu, P. P. S., Shah, C., and Suciu, D. (2026). Claimdb: A fact verification benchmark over large structured data.

Wu, Y., Agarwal, P. K., Li, C., Yang, J., and Yu, C. (2014). Toward computational fact-checking. *PVLDB*, 7(7):589–600.

Yin, P., Neubig, G., Yih, W.-t., and Riedel, S. (2020). TaBERT: Pretraining for joint understanding of textual and tabular data. In *ACL 2020*, pages 8413–8426.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., and Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*, Brussels, Belgium.

Zhang, T., Wang, S., Yan, S., Jian, L., and Liu, Q. (2023). Generative table pre-training empowers models for tabular prediction. In *EMNLP 2023*, pages 14836–14854.