MoreData+: Enriquecimento Semântico para Grandes Volumes de Dados Geolocalizados

Thomas S. M. Chang[1], **Germano B. dos Santos**[1],
Leonardo J. A. S. Figueiredo[2], **Fabrício A. Silva**[1]

[1]Instituto de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa - Campus Florestal (UFV – CAF)
Rodovia LMG 818, km 6, 35.690-000 – Florestal – MG – Brazil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

thomas.chang@ufv.br, germano.santos@ufv.br, fabricio.asilva@ufv.br

leonardo.alves@dcc.ufmg.br

Resumo. *A ferramenta MoreData busca facilitar o trabalho do analista de dados geoespaciais, ao simplificar os processos de enriquecimento semântico que muitas vezes são realizados de maneira manual. Ela não estava preparada para receber um grande volume de dados de maneira eficiente, ao considerar o tempo de execução. Este trabalho apresenta uma nova abordagem ao modelo de dados da MoreData, com o auxílio das bibliotecas GeoPandas e Dask-GeoPandas, para que possam ser utilizadas. Realizou-se uma análise quantitativa das métricas tempo de execução, uso de memória e uso de CPU, para as novas abordagens. Como resultado, foi possível diminuir significativamente o tempo de execução do enriquecimento de dados, e utilizar o MoreData para grandes volumes de dados.*

1. Introdução

Nos últimos anos, houve um aumento considerável na coleta e disponibilidade dos dados geoespaciais que podem ser provenientes, por exemplo, de dispositivos móveis e redes sociais [Domingues et al. 2020]. Por sua vez, estes dados geralmente são representados pela tupla $\langle latitude, longitude, timestamp \rangle$, mas também podem ser representados por uma chave de identificação como $\langle código\_postal \rangle$. Porém, estas informações brutas não possuem um valor semântico, e consequentemente, são ineficazes nas tomadas de decisão. Como exemplo, nas redes sociais, padrões de comportamento geolocalizados podem ser explorados para oferecer um serviço mais eficiente de publicidade.

Com isso, uma das formas de sanar este problema é enriquecendo as bases de dados, para que os usuários possam ter uma melhor compreensão [Lira 2014]. Porém, isso pode ser uma tarefa bastante árdua tendo em vista que existem alguns problemas, como a grande quantidade de dados disponíveis, as diversas fontes para se realizar o enriquecimento, e a realização do trabalho, que muitas vezes é manual. Para mitigar esses problemas, foi proposto o MoreData [Figueiredo et al. 2021], uma ferramenta que visa unir os dados de localização brutos originais a fontes externas com o mínimo de esforço do analista. Sendo assim, dado uma localização representada por $\langle latitude, longitude \rangle$,

ou alguma outra chave de identificação, a ferramenta compara os dados originais com os dados de outras fontes, de acordo com interesse do usuário, retornando um conjunto de dados enriquecido. O componente central do MoreData[1] é o *Enricher*, que é composto por dois módulos principais: o *Connector* responsável pela conexão com a fonte de dados externa e o *Contructor*, responsável por realizar o enriquecimento. Atualmente, a ferramenta é capaz de enriquecer dados geoespaciais utilizando os quatro conectores abaixo:

- *OpenStreetMap*[2]: Coleta dados da API *OpenStreetMap* e indexa localmente os elementos retornados em uma *R-Tree*, estrutura de dados com o objetivo de indexar informações multi-dimensionais [Guttman 1984] para melhorar o tempo da busca geoespacial.
- API: Executa solicitações HTTP (*Hyper Text Transfer Protocol*) a partir de um URI (*Uniform Resource Identifier*) definido pelo usuário para coletar as informações necessárias para a associação dos dados da API com os dados originais.
- Banco de Dados Relacional: Conecta-se a um banco de dados relacional, após isso realiza a consulta através do *script* fornecido pelo usuário, retornando o conjunto de resultados que é utilizado para enriquecer os dados originais.
- *Elastic Search*[3]: O conector para esta fonte utiliza duas estruturas de configuração JSON, a *pipeline*, responsável por definir qual atributo do índice será enriquecido e qual será utilizado como chave para o relacionamento; e política, que define o uso de um campo para enriquecer um índice do Elastic Search.

Até então, o MoreData possui suporte de leitura apenas para arquivos do tipo JSON, que restringe a escalabilidade, porque não está preparado para enriquecer grandes volumes de dados. Isso inviabiliza o tempo necessário para realizar o enriquecimento. Além disso, apesar do formato JSON ser bastante popular, existem novos tipos de dados, como o GeoPandas Dataframe [Jordahl et al. 2022]. O objetivo deste trabalho é estender o MoreData para que contemple duas novas abordagens: GeoPandas e Dask-GeoPandas. Com essas abordagens, será possíevel enriquecer grandes volumes de dados.

A motivação do trabalho se dá tendo em vista que o MoreData é uma ferramenta que ainda é um protótipo e possui melhorias a serem feitas. Assim, o objetivo deste trabalho é melhorar o desempenho do conector *OpenStreetMap* (OSM), no que se diz respeito ao tempo gasto por enriquecimento, processamento e memória utilizada durante o enriquecimento dos dados, para que o mesmo seja mais eficiente no quesito tempo de execução e consequentemente mais atrativo para os usuários. Por fim, também será apresentada uma análise quantitativa dos dados, comparando o uso de memória, o processamento e também o tempo gasto pelo OSM ao enriquecer dados utilizando JSON, GeoPandas Dataframe e Dask-GeoPandas Dataframe.

O restante do texto está organizado da seguinte forma: A seção 2 apresenta o desenvolvimento e as principais modificações realizadas, com a utilização dos materiais e dos métodos. A seção 3 traz os resultados obtidos e sua avaliação. Por fim, as considerações finais são apresentadas na seção 4.

[1]https://github.com/NESPEDUFV/more-data

[2]https://www.openstreetmap.org/#map=4/-16.68/-63.19

[3]https://elasticsearch-py.readthedocs.io/en/7.x/index.html

2. Desenvolvimento

O desenvolvimento foi separado em duas partes. Inicialmente, visando ambientar-se com o projeto, foram resolvidas duas pendências em aberto na versão original. A segunda parte destinou-se para a implementação das melhorias de desempenho, utilizando as bibliotecas GeoPandas e Dask-GeoPandas.

2.1. Correções do MoreData

Para a condução desse projeto, foi necessário compreender o objeto de estudo: a ferramenta MoreData e o seu funcionamento. Além da leitura do artigo original, necessitou-se um entendimento mais profundo do padrão de projeto criacional *Builder*[Gamma et al. 1994] e do padrão de projeto comportamental *Strategy*[Gamma et al. 1994]. Em seguida, realizou-se testes com a ferramenta original para averiguar os resultados gerados.

2.1.1. *Issue*: O *buffer* de um ponto requer grandes recursos no *OSMPlacesConnector*

O módulo *OSMPlacesConnector* pode coletar dados da API *OpenStreetMap* ou utilizar base de dados previamente baixadas do OSM. Após obter os dados, o conector indexa localmente os elementos e retorna uma R-Tree para melhorar o tempo de busca geoespacial. Após isso, o conector enriquece os dados originais com a geometria geoespacial fornecida.

Deste modo o *OSMPlacesConnector* possui a função *'geodesic_point_buffer'* que era executada repetidamente caso fosse realizado mais de um enriquecimento, o que consumia recursos computacionais desnecessários. Dessa forma, a solução foi adicionar uma *flag booleana* como parâmetro, a qual auxilia no controle de fluxo e permite que o *buffer* seja realizado apenas uma vez.

2.1.2. *Issue*: Suporte para vários arquivos usando OMPlacesConnector

O OSMPlacesConnector utilizava o parâmetro 'file' para receber um único arquivo *csv* que auxiliaria no enriquecimento da base de dados original. Porém, caso fosse necessário enriquecer com diferentes tipos de fontes, poderia levar muito tempo para calcular cada enriquecimento porque é necessário criar muitas instâncias de enriquecedores. A solução para esse problema foi criar uma lista de arquivos ao invés de apenas um. Assim foi adicionado o parâmetro *'files'*, que durante a leitura do arquivo possibilita, através de um laço de repetição, que mais de um arquivo seja lido.

2.2. MoreData+

O MoreData+ é um aprimoramento da versão original, e visa tornar mais eficiente a ferramenta, ao mesmo tempo que apresenta novas possibilidades para realizar o enriquecimento.

2.2.1. Materiais e Métodos

Inicialmente criou-se a implementação da classe GeopandasData, e alterou-se o modelo de dados para que o MoreData fosse capaz de continuar utilizando a implementação original e as novas abordagens. Em seguida, criou-se a classe DaskGeopandasData de maneira similar a GeopandasData. Com o intuito de economizar tempo, durante o desenvolvimento utilizou-se uma base de dados presente nos exemplos do MoreData, a do Airbnb da cidade de Berlin[4] com 19858 registros com informações sobre os imóveis.

Depois da implementação das novas classes, se fez necessário uma maneira de comparar o desempenho entre todas as abordagens. Para isso as seguintes métricas foram escolhidas:

- Tempo de execução do enriquecimento;
- Consumo de memória em *Megabytes* (MB), medido pelo consumo total antes do enriquecimento e consumo total exatamente no final do enriquecimento;
- Porcentagem de uso do processador.

Estas métricas foram coletadas através de uma rotina que executou e enriqueceu os dados utilizando o conector para enriquecer dados do OSM, e a relação dos três diferentes tipos de soluções: JSON, GeoPandas e Dask-GeoPandas e a quantidade de registros utilizados é mostrado na Tabela 1. O computador de testes em que o enriquecimento foi realizado é formado por uma CPU Intel Core I5 - 7400, que conta com 4 núcleos, 4 *threads* e frequência máxima de 3,50 GHz. Sua memória RAM é composta por dois módulos de 8 GB e 2666 MHz, totalizando 16 GB disponíveis.

Após recolher as métricas, realizou-se a análise utilizando um notebook no Google Colab[5], e as principais bibliotecas utilizadas foram a Pandas [6] e a Seaborn [7].

Registros	JSON	GeoPandas	Dask-GeoPandas
1.000	X	-	-
5.000	X	-	-
10.000	X	-	-
50.000	X	-	-
100.000	X	X	X
200.000	X	-	-
250.000	-	X	X
400.000	X	-	-
500.000	-	X	X
800.000	-	X	X
1.000.000	-	-	X
1.800.000	-	-	X

Tabela 1. Relação entre a quantidade de registros e a solução do enriquecimento.

[4]http://insideairbnb.com/get-the-data/
[5]https://colab.research.google.com
[6]https://pandas.pydata.org/
[7]https://seaborn.pydata.org/

2.2.2. Implementação do *GeopandasData*

Anteriormente o conector do *OpenStreetMap* aceitava apenas arquivos do tipo JSON, o que limitava as possibilidades de aplicação da ferramenta. Com o intuito de aumentar o desempenho e estender a compatibilidade com diferentes tipos de arquivos, modificou-se o modelo de dados, inicialmente nomeado como Data, apresentado na Figura 1. Essa passou a ser uma classe mãe para a subclasse GeopandasData que possui dois métodos: o 'from_geodataframe(cls, geodataframe)' que retorna um tipo GeopandasData a partir do geodataframe passado para o método e o 'from_path(cls, path)' que também retorna um tipo GeopandasData, porém lê o arquivo a partir do caminho passado para o método pelo o usuário. Esses detalhes podem ser visualizados esquematicamente na Figura 2.

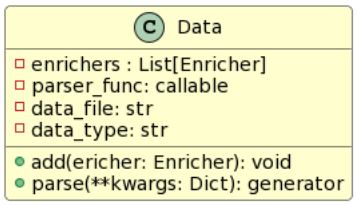

Figura 1. Modelo de dados antigo.

Após esta refatoração houve a necessidade de alterar o código do conector OSM-PlacesConector, que passou a ser capaz de utilizar arquivos *csv* transformados em GeoPandas Dataframe. Agora o método *enrich()* realiza a verificação se a instância do tipo de arquivo recebido é JsonData ou GeopandasData. Após essa verificação, o *enrich()* deve retornar o método *enrichJsonData*, que realiza o enriquecimento da maneira original, ou o método *erinchGeopandasData*, que realiza um *buffer* da geometria e em seguida retorna um *spatial join* com os dados a serem enriquecidos.

2.2.3. Implementação do DaskGeopandasData

Para realizar essa funcionalidade foi necessário instalar como dependência o pacote Dask-GeoPandas, que por ser uma ferramenta recente não possui todas as funcionalidades que a GeoPandas.

Dito isso, a implementação foi semelhante ao do GeoPandas, onde necessitou-se criar uma nova subclasse para o modelo de dados Data, como mostra a Figura 2. A diferença entre as duas subclasses está na função que realiza o *spatial join*, que devido a uma limitação do Dask-GeoPandas, não possui para o parâmetro *how* o tipo *left*, que utiliza todas as chaves do DataFrame original para retornar os dados enriquecidos. Sendo assim utilizou-se o único tipo disponível para o DaskGeopandasData, o tipo *inner*, que faz a interseção entre as chaves das duas bases de dados para retornar os dados enriquecidos.

É necessário salientar que para o enriquecimento acontecer de maneira paralela, o usuário necessita estar utilizando um *Dask Cluster* e adicionar ao método *enrich()* a chamada *compute()*.

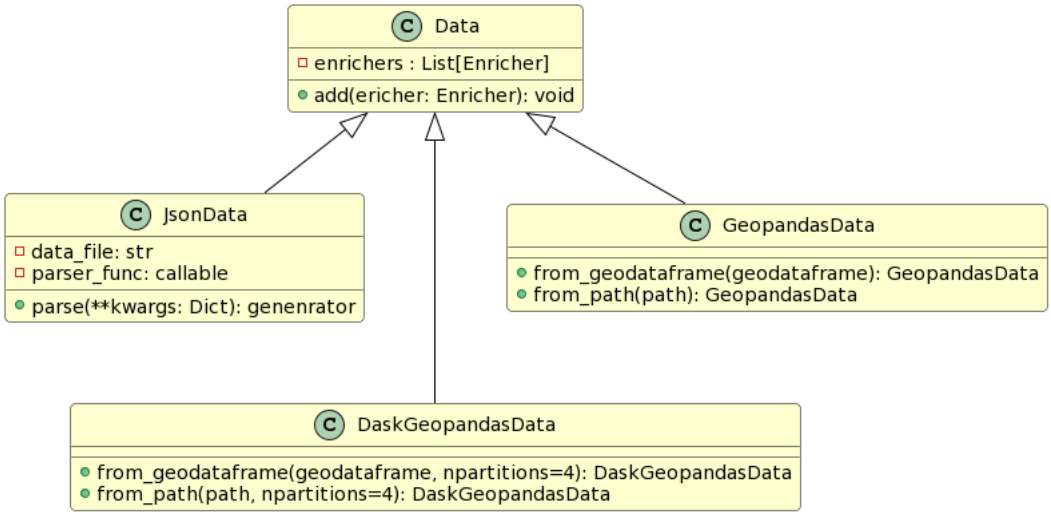

Figura 2. Novo modelo de dados.

3. Análises e Resultados

Para os testes de desempenho utilizou-se uma base de dados com registros da cidade de São Paulo, coletados de 31 de Dezembro de 2019 até 01 de Março de 2022, e fornecidos por uma empresa parceira. Os registros possuem os seguintes campos: *h3_hash*, *app_user_id*, *date_time cod_municipio*, latitude e longitude. Essa base original foi enriquecida com dados de turismo da mesma cidade, com uma base de dados que possui mais de 270 colunas. O principal dado é o *geometry*, que nos dirá se o registro enriquecedor estará dentro ou não da área de interesse dada pelo o usuário. Os demais campos são características do local como nome, endereço e telefone.

Afim de avaliar o desempenho em cada um dos cenários, primeiro criou-se um *script* que executou o enriquecimento para arquivos do tipo JSON uma vez para cada quantidade de registros apresentadas na Tabela 1. Em seguida, armazenou-se em um arquivo *csv* a quantidade de registro, a diferença de memória antes do enriquecimento e depois do enriquecimento, o tempo gasto para o enriquecimento e por fim, o consumo de CPU. O mesmo foi feito para o GeoPandas e Dask-GeoPandas porém, esses foram executados 8 vezes devido ao tempo de execução ser significativamente menor. Após a coleta, criou-se um *notebook* no Google Colab[8], e as bases de dados foram importadas e tratadas devidamente para gerar os gráficos apresentados nas análises de desempenho.

3.1. JSON e GeoPandas

Ao observar o gráfico da Figura 3 percebe-se que os arquivos do tipo JSON tendem a manter constante o uso de memória de 1.000 a 200.000 registros. Já entre 200.000 e 400.000 observa-se um crescimento no uso da memória. Não foi possível enriquecer mais do que essa quantidade tendo em vista que o tempo de execução ultrapassaria as 16 horas, tornando inviável para o experimento. Por outro lado os GeoPandas DataFrame tem o consumo de memória mais elevado. Entre 100.000 e 500.000 registros o uso de memória teve um crescimento linear, já entre 500.000 e 800.000 o coeficiente linear é maior, porém continua sendo linear, esse tipo de dado começou a ser coletado a partir

[8]https://colab.research.google.com/drive/1v6Am6tmMnqB9asGCalyxVCr0N3Xig5hQ?usp=sharing

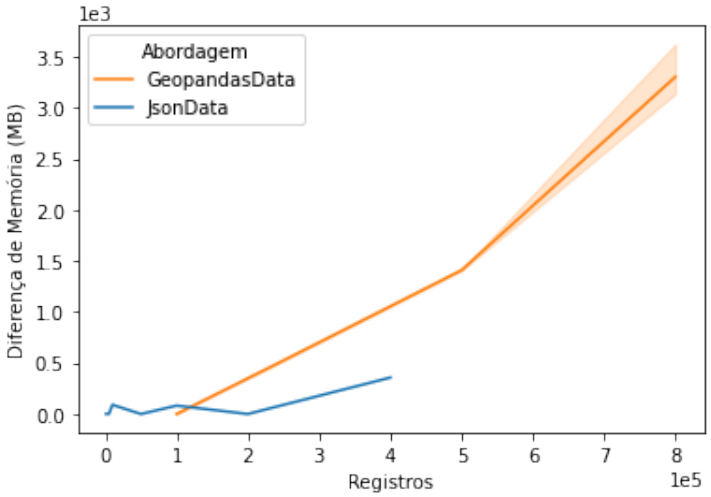

Figura 3. Comparação entre JSON e GeoPandas da diferença entre o consumo de memória antes e depois do enriquecimento em função da quantidade de registro.

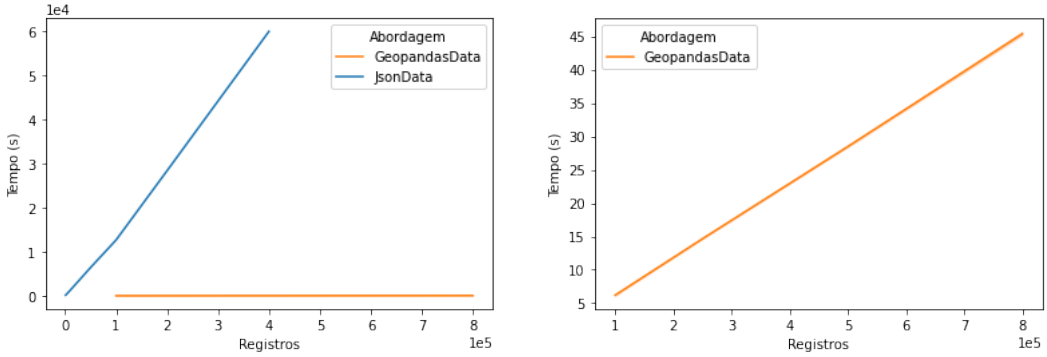

Figura 4. Comparação entre JSON e GeoPandas do tempo gasto para o enriquecimento em função da quantidade de registro.

dos 100.000 registros porque abaixo disso a execução era muito rápida e consumia pouca memória, e isso para a comparação não traria benefícios. Vale evidenciar que o baixo consumo de memória do JSON ocorre devido os arquivos serem lidos em partes, o que não necessita de muita memória. Porém, isso leva a um tempo significativamente maior de execução, como será mostrado a seguir.

Já para o tempo de execução do enriquecimento, apresentado no gráfico da Figura 4, o cenário se inverte. Os arquivos *csv* transformados em GeoPandas DataFrame ficam abaixo dos 60 segundos entre 100.000 e 800.000 registros. Já os arquivos JSON crescem linearmente e 400.000 registros levam aproximadamente 60.000 segundos para executar.

Logo, pode-se afirmar que o desempenho para os arquivos *csv* transformados em GeoPandas DataFrame é muito superior ao JSON em termos de tempo de execução. Porém, deve-se ficar atento ao consumo de memória, que é consideravelmente maior comparado ao JSON. Dependendo da quantidade de registros pode ocorrer o 'estouro' da memória.

3.2. GeoPandas e Dask-GeoPandas

Para realizar a execução do Dask-GeoPandas configurou-se o cluster da seguinte forma: *n_workers = 3* atribuindo o número de núcleos para 3; *threads_per_worker = 1* atribuindo uma thread para cada núcleo do processador; *memory_limit = '5gb'* atribuindo a quantidade máxima de memória para cada núcleo.

Dito isso, ao analisar o gráfico da Figura 5 é possível perceber que o consumo de memória do GeoPandas é significativamente maior do que o Dask GeoPandas. Pode-se afirmar que o Dask-GeoPandas utiliza aproximadamente a mesma quantidade de memória para 1.800.000 que o GeoPandas utiliza para 800.000. Não foi possível enriquecer mais dados porque a memória excedeu o seu limite após essa quantidade de registros para cada uma das abordagens. Ao analisar o gráfico da Figura 6, que apresenta o tempo de

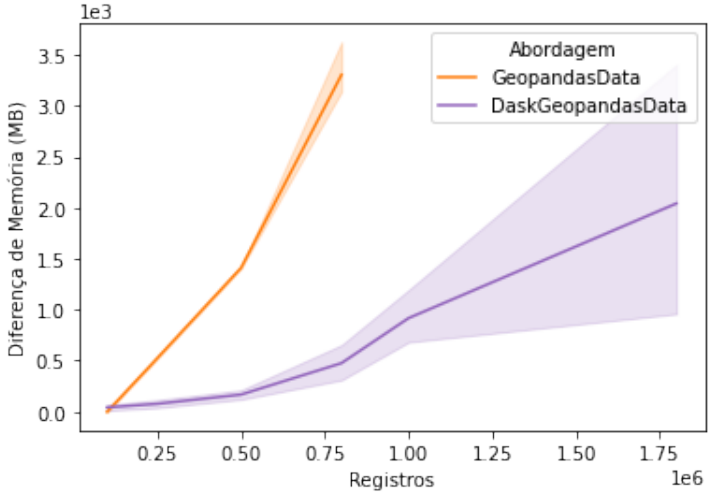

Figura 5. Comparação entre GeoPandas e Dask-GeoPandas em relação a diferença entre o consumo de memória antes e depois do enriquecimento em função da quantidade de registro.

execução do enriquecimento e função da quantidade de registros, é possível perceber que o tempo de execução do Dask-GeoPandas foi menor, tendo assim um desempenho melhor em relação ao GeoPandas. Enquanto o GeoPandas demora aproximadamente 50 segundos para enriquecer 800.000 registros, o Dask-GeoPandas consegue enriquecer 1.300.000 no mesmo tempo. Vale destacar que com o Dask-GeoPandas foi possível enriquecer no computador de testes até 1.800.000 registros em 59 segundos, já com o GeoPandas foi possível chegar a no máximo 800.000 registros. Isso mostra a capacidade de escalabilidade do Dask-GeoPandas.

Porém ao analisar o consumo de CPU na Figura 7, percebe-se que o Dask-GeoPandas consome praticamente o dobro de recurso em comparação com o GeoPandas, quando o enriquecimento é executado. Isso é esperado, tendo em vista que a função do Dask é processar paralelamente os dados, consequentemente utilizando mais CPU.

4. Conclusão e Trabalhos Futuros

Este trabalho buscou viabilizar o enriquecimento semântico para grandes volumes de dados geoespaciais no MoreData, tendo em vista que anteriormente o enriquecimento para

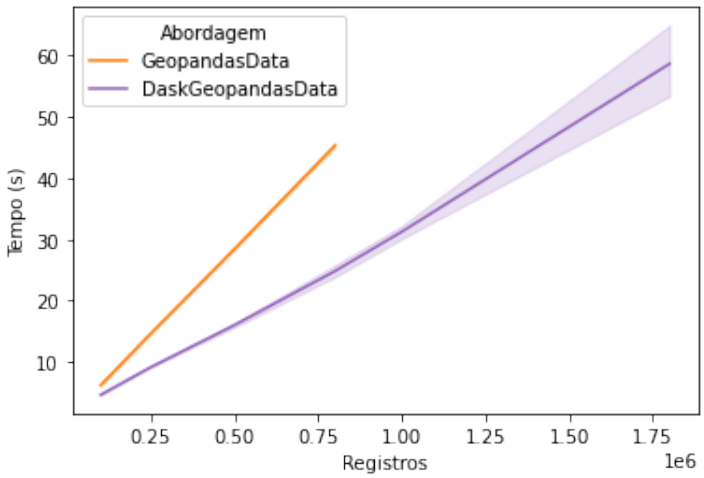

Figura 6. Comparação entre GeoPandas e Dask-GeoPandas do tempo gasto para o enriquecimento em função da quantidade de registro.

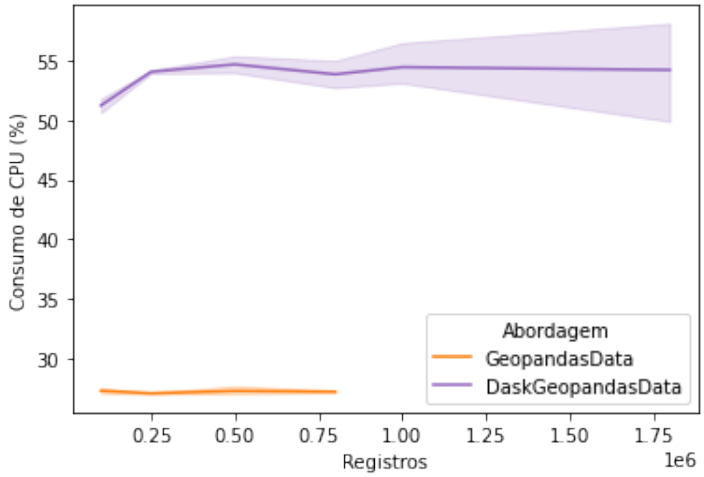

Figura 7. Comparação entre GeoPandas e Dask-GeoPandas em relação ao consumo de CPU antes e depois do enriquecimento em função da quantidade de registro.

tal volume era inviável, no quesito tempo. Sendo assim, foi proposta uma nova abordagem, que refatorou e adicionou dois novos modelos de dados o GeopandasData e o Dask-GeopandasData. Esses utilizaram o auxílio das bibliotecas GeoPandas e Dask-GeoPandas para realizarem o enriquecimento semântico de maneira mais rápida que a original.

Após o estudo, pode-se concluir que as alterações realizadas neste projeto, contribuíram de modo expressivo com a melhora do tempo de execução e também a quantidade de registros que a MoreData é capaz de enriquecer por arquivo. Porém, deve ser usado de maneira cautelosa para não exceder o consumo de memória disponível pela maquina.

Para trabalhos futuros há a possibilidade de adicionar o *Dask* no *Functional Region Connector*, como também há possibilidade de adicionar o enriquecimento paralelo aos outros conectores, tendo em vista que existem conectores que não possuem de-

pendência e podem ser executados em paralelo.

Referências

Domingues, A., Silva, F., Santos, L., Souza, R., Coimbra, G., and Loureiro, A. A. F. (2020). Dados geoespaciais: Conceitos e técnicas para coleta, armazenamento, tratamento e visualização. *Sociedade Brasileira de Computação*.

Figueiredo, L. J. A. S., dos Santos, G. B., Souza, R. P. P. M., Silva, F. A., and Silva, T. R. M. B. (2021). Moredata: A geospatial data enrichment framework. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '21, page 419–422. Association for Computing Machinery, New York, NY, USA.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition.

Guttman, A. (1984). R trees: A dynamic index structure for spatial searching. volume 14, pages 47–57.

Jordahl, K., den Bossche, J. V., Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., Rey, S., Ren, C., Arribas-Bel, D., Wasser, L., Wolf, L. J., Journois, M., Wilson, J., Greenhall, A., Holdgraf, C., Filipe, and Leblanc, F. (2022). geopandas/geopandas: v0.11.0.

Lira, M. A. B. d. (2014). Uma abordagem para enriquecimento semântico de metadados para publicação de dados abertos. *Dissertação (mestrado) - UFPE, Centro de Informática, Programa de Pós-graduação em Ciência da Computação, 2014.*