Pré-processamento de artigos do projeto ar5iv para a Recuperação de Informação Matemática

Filipe Parisoto Ribeiro[1], **Flavio Barbieri Gonzaga**[1]

[1]Instituto de Ciências Exatas - Ciência da Computação
Universidade Federal de Alfenas (UNIFAL-MG)

filipe.ribeiro@sou.unifal-mg.edu.br, flavio.gonzaga@unifal-mg.edu.br

Abstract. *Around the year* 2 007*, efforts began on what, 15 years later in* 2 022*, would become the ar5iv project. With the goal of making scientific papers (in LaTeX format) more accessible in a user-friendly way on the web, the project proposes the conversion of the arXiv preprint repository to the HTML5 format. Still in* 2 022*, the ar5iv project announced that it was able to successfully convert about* 74% *of the entire arXiv. In light of the project's results, an old demand from the Mathematical Information Retrieval area arises: how to search for mathematical formulas in scientific papers? This paper presents the development of two main algorithms capable of downloading all ar5iv pages, and then pre-processing, persisting in a database, and finally, extracting all formulas in MathML Presentation format. The proposed solution extracted over* 1 800 000 *articles and almost* 300 000 000 *distinct formulas.*

Keywords: *arXiv, ar5iv, MathML, HTML5, Math Information Retrieval.*

Resumo. *Por volta do ano de* 2 007 *tiveram início os esforços no que, 15 anos mais tarde, em* 2 022*, viria a ser o projeto ar5iv. Com o objetivo de tornar os artigos científicos (em formato LaTeX) acessíveis de maneira mais amigável na web, o projeto propõe a conversão do repositório de pré-publicações arXiv para o formato HTML5. Ainda em* 2 022 *o projeto ar5iv anunciou ter sido apto em converter, de maneira bem sucedida, cerca de* 74% *de todo o arXiv. Diante dos resultados do projeto, uma antiga demanda da área de Recuperação de Informação Matemática vem a tona: como buscar por fórmulas matemáticas presentes em artigos científicos? O presente trabalho apresenta o desenvolvimento de dois algoritmos principais, capazes de fazer o download de todas as páginas da ar5iv, e em seguida, pré-processar, persistir em banco de dados, e finalmente, extrair todas as fórmulas em formato MathML Presentation. A solução desenvolvida extraiu mais de* 1 800 000 *artigos e quase* 300 000 000 *de fórmulas distintas.*

Palavras-chave: *arXiv, ar5iv, MathML, HTML5, Recuperação de Informação Matemática.*

1. Introdução

Criada em $1\,991$ por Paul Ginsparg, o arXiv[arXiv 1991] é um repositório de pré-publicações de artigos científicos (antes do processo de revisão por pares) que, atualmente, hospeda mais de $2,2$ milhões de artigos das seguintes áreas: Física, Matemática, Ciência da Computação, Biologia Quantitativa, Finança Quantitativa, Estatística, Engenharia Elétrica, Ciência de Sistemas e Economia. Em virtude do grande volume de conhecimento, o arXiv desperta o interesse de indexação para busca e recuperação de informação nas mais variadas ferramentas de busca pela Internet.

O processo de indexação consiste na criação de índices que facilitem a recuperação de informação relevante em meio a uma grande coleção de dados [Baeza-Yates and Ribeiro-Neto 2013]. Apesar da diversidade de técnicas de análise dos documentos a serem indexados, o foco principal é a extração de palavras (termos-chave) que devem compor o índice. Dessa forma, o índice contém o conjunto de todas as palavras extraídas de uma coleção de documentos, de modo que, dada uma consulta submetida por um usuário, é possível recuperar rapidamente quais os documentos possuem as palavras contidas na consulta.

Considere o exemplo a seguir:

Documentos:

1. "A área da Ciência da Computação é promissora."
2. "A Computação possui diversas sub-áreas."
3. "A palavra ciência significa conhecimento."

Índice (palavras $\rightarrow$ lista de documentos):

"a" $\rightarrow \{1,2,3\}$
"área" $\rightarrow \{1\}$
"da" $\rightarrow \{1\}$
"ciência" $\rightarrow \{1,3\}$
"computação" $\rightarrow \{1,2\}$
"é" $\rightarrow \{1\}$
"promissora" $\rightarrow \{1\}$
"possui" $\rightarrow \{2\}$
"diversas" $\rightarrow \{2\}$
"sub" $\rightarrow \{2\}$
"áreas" $\rightarrow \{2\}$
"palavra" $\rightarrow \{3\}$
"significa" $\rightarrow \{3\}$
"conhecimento" $\rightarrow \{3\}$

Dado o índice acima, caso um usuário submetesse uma consulta como sendo: "ciência da computação", esperando receber documentos contendo todas as palavras, fica claro que somente o documento 1 seria retornado. Isso porque a palavra "ciência" aparece nos documentos 1 e 3, a palavra "da" aparece apenas no documento 1, e por fim, a palavra "computação" está presente nos documentos 1 e 2. Dessa forma, apenas o documento 1 possui todas as palavras da consulta.

Neste simples exemplo fica claro que diversas técnicas e tratamentos poderiam ser aplicados de modo a se melhorar a busca proposta. Por exemplo, as palavras “áreas” e “área” deveriam ser tratadas como uma única entrada no índice, dentre outras melhorias possíveis.

Entre os principais métodos de extração de palavras se encontram técnicas de análise semântica, análise de variação de palavras e análise de relevância, e ao combinar estes itens, é possível criar um índice para facilitar a identificação dos documentos relevantes para uma dada pesquisa, em meio a uma coleção de dados[Manning 2008].

Com relação ao repositório arXiv, tema do presente trabalho, indexá-lo era uma tarefa complexa até pouco tempo atrás. O principal motivo é que os artigos disponíveis estão em sua maioria no formato PDF. Por se tratarem de artigos científicos, tais arquivos possuem elementos como imagens, fórmulas matemáticas, tabelas, gráficos, entre outros itens que dificultam consideravelmente o processo de recuperação da informação contida nos mesmos. Embora na maioria dos casos seja possível também acessar o arquivo fonte dos artigos (em formato LaTeX, doc, dentre outros), ainda assim, o processo de extração dos elementos de maneira limpa (livre de marcações), não era algo simples.

Contudo, recentemente, com a criação do projeto ar5iv [ar5iv 2022], a complexidade da tarefa de indexação reduziu, visto que o projeto em questão tem por objetivo a transcrição dos artigos contidos no arXiv para o formato HTML5. O que permite que ferramentas de busca possam processar as páginas geradas referentes aos artigos e extraírem informações relevantes para a criação dos índices do processo de busca e recuperação de informação.

Dessa forma, o objetivo do presente trabalho é desenvolver uma solução que seja capaz de realizar o processo de *crawling* (*download* dos arquivos) do ar5iv, tratamento e persistência dos mesmos. Em virtude do ar5iv ter bastante conteúdo voltado para as Ciências Exatas, serão devidamente extraídas também as fórmulas matemáticas dos artigos. Dessa forma, o trabalho proposto pode ser posteriormente utilizado por ferramentas de busca (textual e matemática) que tenham o interesse de indexar o conteúdo proveniente do repositório ar5iv. A seguir apresenta-se com detalhes os objetivos do trabalho.

1.1. Objetivos

Temos como objetivo deste trabalho: construir uma solução para pré-processamento das páginas referentes aos artigos do ar5iv. Sendo os objetivos específicos:

- Construção de uma aplicação para *download* das páginas geradas pela conversão do ar5iv;
- Construção de uma aplicação para processamento das páginas obtidas;
- Extração e armazenamento de todas as fórmulas durante o processamento;
- Extração do conteúdo textual durante o processamento.

2. Trabalhos Relacionados

Esta seção tem por objetivo relacionar trabalhos desenvolvimentos anteriormente que reforçam não só a complexidade como também o desejo pela indexação de repositórios como o arXiv para a Busca e Recuperação Matemática.

O artigo [Hu et al. 2013] apresenta o WikiMirs, uma ferramenta criada para facilitar a busca matemática em páginas presentes na Wikipedia. O sistema proposto é capaz

de fazer busca textual e busca por fórmulas em uma coleção com 13 milhões de páginas, as quais somam 495 958 fórmulas. Para tanto, as estruturas matemáticas em LaTeX identificadas foram extraídas e pré-processadas, e em seguida criado o índice que facilita a busca.

Já o trabalho [Sojka 2012] demonstra a capacidade do MIaS(Math Indexer and Searcher) em indexar artigos presentes no arXiv e oferecer uma busca baseada não só em palavras, como também em estruturas matemáticas. Porém, o trabalho apresentou problemas de escalabilidade quando a base indexada superou os 400 000 artigos do arXiv. No total o projeto indexou 438 000 documentos pré-processados do arXiv com 158 milhões de fórmulas identificadas.

Temos também um trabalho de extração de artigos e fórmulas matemáticas diretamente do arXiv, no formato TeX. [Rodrigues and Gonzaga 2019] apresentaram uma solução que foi capaz de extrair, no período de um ano, 136 064 artigos contendo fórmulas matemáticas que totalizam 22 346 623 fórmulas distintas. Esta solução teve todo seu processamento baseado nos arquivos fonte (no formato TeX) dos artigos publicados no arXiv.

O presente trabalho trás uma nova solução de extração de artigos e fórmulas provenientes das páginas do projeto ar5iv. Sendo esta solução mais eficiente no processo de extração, o que trás melhorias perante [Rodrigues and Gonzaga 2019], pois neste trabalho o processamento é feito sob páginas HTML, e não mais arquivos TeX. O resultado deste projeto também apresenta pontos de evolução para a ferramenta WikiMirs [Hu et al. 2013], que pode estudar a viabilidade de se indexar não só o Wikipedia, mas também a ar5iv, utilizando-se dos resultados gerados por este trabalho. O projeto MIaS [Sojka 2012] também pode se beneficiar dos resultados aqui obtidos, pois o mesmo apresentou problemas de escalabilidade ao superar o número de 400 000 artigos indexados do arXiv, enquanto nesta solução foi possível a extração de um número superior de artigos e fórmulas.

3. Tecnologias Utilizadas

O escopo desta seção é a contextualização tecnológica do trabalho desenvolvido.

3.1. MathML

O MathML (*Mathematical Markup Language*) é uma linguagem de marcação baseada em XML para descrever notações matemáticas e capturar tanto a sua estrutura quanto seu conteúdo. [Mozilla Contributors 2023b].

A linguagem possui duas variações, o MathML *Presentation* e o MathML *Content*. O MathML *Presentation* preserva a estrutura de notação de maneira abstrata facilitando a renderização em diferentes mídias, garantindo que uma dada marcação reproduza sempre a mesma fórmula visualmente, independente do tamanho ou tecnologia utilizada para visualização. O MathML *Content* preserva a estrutura matemática de forma a facilitar com as aplicações sejam capazes de inferir o significado matemático por trás da estrutura. A Tabela 1 apresenta uma comparação de estruturação entre as variações para a fórmula $ax^2 + bx + c$.

Tabela 1. Comparativo entre a estruturação *Presentation* e *Content*

MathML *Presentation*

```
<math>
    <mrow>
        <mi>a</mi>
        <msup>
            <mi>x</mi>
            <mn>2</mn>
        </msup>
        <mo>+</mo>
    </mrow>
    <mrow>
        <mi>b</mi>
        <mi>x</mi>
        <mo>+</mo>
    </mrow>
    <mrow>
        <mi>c</mi>
    </mrow>
</math>
```

MathML *Content*

```
<math>
    <apply>
        <plus/>
        <apply>
            <times/>
            <ci>a</ci>
            <apply>
                <power/>
                <ci>x</ci>
                <cn>2</cn>
            </apply>
        </apply>
        <apply>
            <times/>
            <ci>b</ci>
            <ci>x</ci>
        </apply>
        <ci>c</ci>
    </apply>
</math>
```

3.2. Banco de dados relacional

Um banco de dados baseado no modelo relacional, proposto por EF Codd em 1970 [Codd 1970], tem seus dados armazenados em tabelas, e internamente a tabela os dados são organizados em colunas que possuem seus mais variados tipos de representação (inteiro, textos, data, números reais, etc). Relacionadas uma as outras, por meio de ao menos uma coluna em comum, as tabelas são capazes de representar cenários do mundo real. Um exemplo: com uma tabela de "clientes", com as colunas "código" e "nome"; e uma tabela "pedidos", com "código", "descrição" e "código_cliente"; é possível mapear quem são os clientes, e quais são seus pedidos e armazenar todos estes dados para posterior recuperação.

3.2.1. MySQL

O *MySQL* é classificado como um SGBD (Sistema de Gerenciamento de Banco de Dados) relacional. Sendo um servidor Open Source, o *MySQL* fornece um conjunto de interfaces para operações de inserção, recuperação, alteração e remoção de dados em bases as quais ele gerencia, assim como também fornece suporte para conexão de outras aplicações para consumo dos dados nele presentes [MySQL 2023].

3.2.2. MariaDB

Assim como o *MySQL*, o *MariaDB* é um sistema para gerenciamento de banco de dados, tendo sido criado utilizando o próprio *MySQL* como base. Logo, ambos apresentam semelhanças no funcionamento assim como um conjunto de operações leitura/escrita e conectividade muito similares [MariaDB Corporation 2023].

3.3. HTTP

HTTP (*Hypertext Transfer Protocol*) é um protocolo utilizado para trasferência de hypertextos entre aplicações e servidores [Fielding et al. 1999].

3.4. Driver JDBC

As aplicações Java deste projeto possuem dependência direta com APIs baseadas no modelo JDBC (*Java Database Connectivity*). Este modelo de API oferece ferramentas para conexão e interação com banco de dados relacionais. Para tanto são necessários drivers JDBC, que permitem a comunicação da aplicação para com os servidores de banco de dados [Oracle Corporation 2023].

3.5. Framework Spring

O *Framework Spring* é um conjunto de soluções genéricas que oferece suporte para criação de aplicações em Java de forma simples e rápida [Johnson et al. 2004]. Basicamente o *Spring* tem por objetivo facilitar o dia-a-dia dos profissionais de desenvolvimento, oferecendo conjuntos de soluções completos e modularizáveis para os mais diversos problemas do mundo real. Nas sub-seções seguintes são apresentados alguns destes módulos:

3.5.1. Spring Boot

Módulo responsável pela automatização das configurações de uma aplicação Java. O mesmo foi criado para facilitar o uso do *Framework* e gerenciar os demais módulos internos à aplicação[Webb et al. 2023]. Neste trabalho, o *Spring Boot* fez-se necessário para auxílio na injeção de dependência, assim como gerenciar os demais módulos utilizados.

3.5.2. Spring Data JPA

Módulo que oferece suporte para integrações com bancos de dados. O mesmo utiliza o *JPA*(*Jakarta Persistence API*) do *Hibernate*, outro grande *Framework* para manipulação de bancos de dados [Gierke et al. 2023]. O uso do *Spring Data JPA* foi crucial para o gerenciamento de conexões com o banco de dados, assim como facilitou a interação com a base.

3.5.3. Spring Cloud OpenFeign

Módulo de integração para o uso do *Feign*, biblioteca desenvolvida para criação de clientes HTTP. [The Spring Team 2023]. O módulo foi utilizado para automatizar o processo de requisições HTTP feitas para a ar5iv.

3.6. HTML e HTML5

HTML é uma linguagem de marcação de hipertexto utilizada para criação de websites e páginas web. O HTML é basicamente responsável pela estruturação dos elementos e conteúdos de uma página. O HTML5 trata-se apenas de uma versão da linguagem de marcação HTML[Mozilla Contributors 2023a].

3.7. Jsoup

Jsoup é uma biblioteca Java que disponibiliza uma API para extração e manipulação de páginas HTML [Hedley 2009]. Neste trabalho a *Jsoup* facilitou a extração das fórmulas matemáticas e manipulação dos arquivos HTML.

4. Arquitetura do Sistema

A arquitetura do sistema desenvolvido é composta por uma base de dados, e duas aplicações Java, cada qual com responsabilidade distinta. Ademais estão descritas, em detalhes, as especifidades dos itens citados.

4.1. Base de dados

Para construção da base de dados foi utilizado o modelo de banco de dados relacional, sendo o *MySQL* e *MariaDB* os servidores de banco de dados escolhidos para esta arquitetura. O motivo da utilização dos dois servidores se deve ao fato de que uma etapa do trabalho foi executada no LaReS (**La**boratório de **Re**des de computadores e **S**istemas distribuídos) da UNIFAL-MG, onde o SGBD *MariaDB* é comumente utilizado; enquanto que a etapa seguinte foi executada no ambiente do Google Cloud, onde o *MySQL* é utilizado.

A base de dados contou com duas tabelas: a primeira, de nome *oai2*, já havia sido previamente populada por pesquisadores do LaReS, tendo sido portanto, apenas incorporada ao trabalho; enquanto que a segunda tabela, de nome *tb_content*, foi proposta no presente projeto. Os campos, juntamente com os tipos de dados e as respectivas descrições são apresentados a seguir.

- Tabela *oai2*:
  - *id* (texto): identificador;
  - *identifier* (texto): identificador do artigo no arXiv;
  - *datestamp* (data): data de publicação;
  - *setSpec* (texto): área de conhecimento;
  - *created* (data): data de criação do registro;
  - *updated* (data): data de atualização do registro;
  - *authors* (texto): autores;
  - *title* (texto): título;
  - *categories* (texto): categoria;
  - *report_no* (texto): número de publicação atribuído à sua instituição;
  - *journal_ref* (texto): caso o artigo já tenha sido publicado em alguma revista ou conferência;
  - *doi* (texto): identificador de objeto digital;
  - *abstract*(texto): resumo;
  - *comments* (texto): número de páginas e número de figuras;

  - *download_result* (numérico): estado de *download*, onde 0 indica que o registro não tem informação de *download*, 1 indica que o HTML5 do artigo foi baixado e salvo, 2 indica que houve erro no processo de *download* e por fim, o 3 indica que o HTML5 do artigo não foi baixado;
  - *index* (numérico): número de índice interno.

- Tabela *tb_content*:
  - *id* (texto): identificador;
  - *co_src* (bytes): conteúdo obtido pós processamento do HTML5 do artigo.

Os campos que compõem a tabela *oai2* representam dados dos artigos presentes no arXiv que foram obtidos por meio da API[1], disponibilizada juntamente à sua documentação, pelo repositório. A documentação conta com uma seção[2] que detalha alguns dos campos descritos acima.

A Figura 1 apresenta a estrutura das tabelas, quanto ao nome dos campos citados e sua forma de armazenamento.

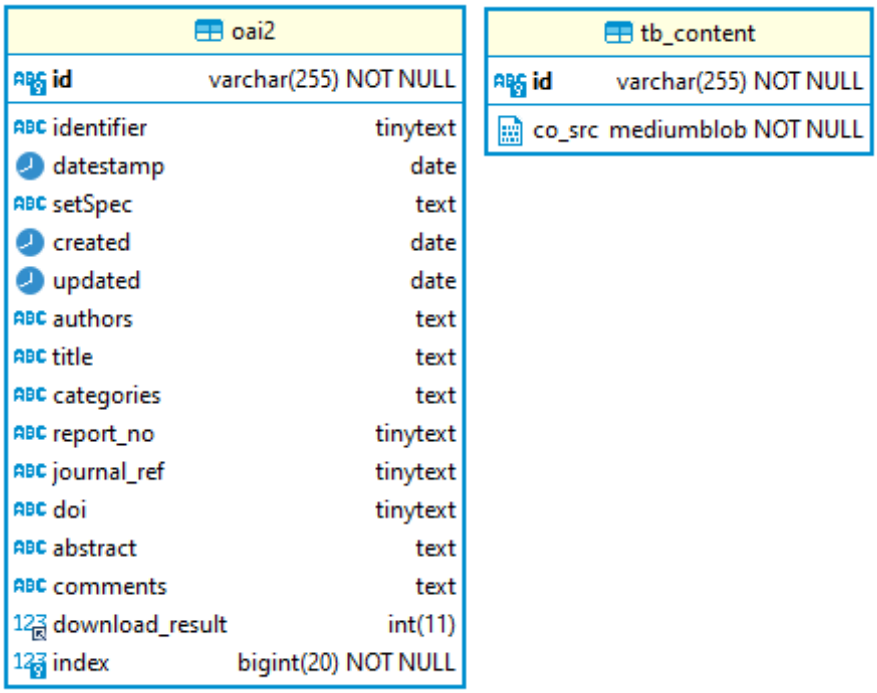

Figura 1. Modelo Relacional da base de dados.

[1] https://info.arxiv.org/help/api/basics.html
[2] https://info.arxiv.org/help/prep.html

A base construída na estrutura detalhada acima possuí em sua totalidade $2\,068\,457$ de registros na tabela *oai2*, os quais armazenam os metadados dos artigos do arXiv.

4.2. Aplicação Java para *Download*

Considerando então, conforme citado na seção 4.1, o fato de que a tabela *oai2* já se encontrava populada com os metadados dos artigos, o primeiro passo do trabalho consistiu no desenvolvimento de uma aplicação utilizando-se do *Framework Spring* que fosse capaz de realizar o *download* de todos os artigos.

No contexto desta aplicação foram utilizados o *Spring Boot, Spring Data JPA*, *Spring Cloud OpenFeign* e o *Driver JDBC MariaDB*.A mesma foi construída partindo da seguinte lógica:

1. Ao iniciar, a aplicação recupera via arquivo de configuração o número base de *threads*, o tamanho do bloco de leitura para cada *thread*, o tempo de espera para novas requisições ao servidor, o valor de estado de *download* a ser consultado, seguindo as definições apresentadas para o campo na seção 4.1, e a *URL* base para requisição e obtenção dos artigos convertidos em formato HTML5;
2. Tomando como base o número total de artigos contidos na tabela *oai2*, cada thread é então responsável por processar um intervalo desta tabela;
    - O tamanho dos intervalos é portanto, definido pelo número de artigos na tabela, divido pelo número de *threads*;
3. Para cada registro da tabela *oai2* (obtido pelas *threads* em blocos), uma requisição foi feita ao servidor do ar5iv;
    - A *URL* da requisição é composta como sendo a *URL* base + id do artigo;
    - É importante frisar que nem todo artigo do arXiv já se encontra convertido em HTML5 no ar5iv, de modo que o retorno da requisição pode ser como arquivo não encontrado;
4. Caso a requisição seja válida, a mesma recebe o arquivo HTML5 do artigo;
    - Como o objetivo do presente trabalho é também permitir futuramente a indexação de fórmulas matemáticas, verifica-se pela existência das mesmas buscando-se pela ocorrência de *tags* do tipo *math* no conteúdo da página;
5. Para artigos obtidos que se encaixem no filtro acima, são criados arquivos HTML e seus conteúdos escritos nos mesmos;
    - O nome dos arquivos são o *id* dos mesmos no arXiv;
6. Após a escrita do arquivo, o registro é atualizado na base de dados para indicar que seu HTML foi salvo;
7. Artigos que não se encaixam no filtro, ou não possuem conversão para HTML5 ainda, são ignorados para criação e escrita de arquivos, mas tem seus registros atualizados na base de dados para indicar processamento sem persistência do HTML;
8. Após uma iteração de requisição, há um tempo de espera parametrizado, para evitar que o servidor do ar5iv sofra com muitas requisições simultâneas;

Ao fim do processamento da aplicação, espera-se que todos os artigos presentes na base tenham sido consultados no servidor do ar5iv. Para aqueles cuja requisição foi bem sucedida, e o conteúdo possui fórmulas matemáticas, o arquivo HTML deve estar salvo localmente na máquina. Estes arquivos compõem a entrada da segunda aplicação.

As Figuras 2 e 3 ilustram o fluxo descrito acima, onde as etapas do mesmo estão destacadas em vermelho em ambas as imagens.

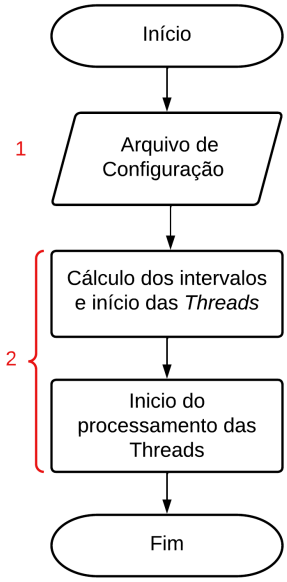

Figura 2. Lógica da *Thread* Principal

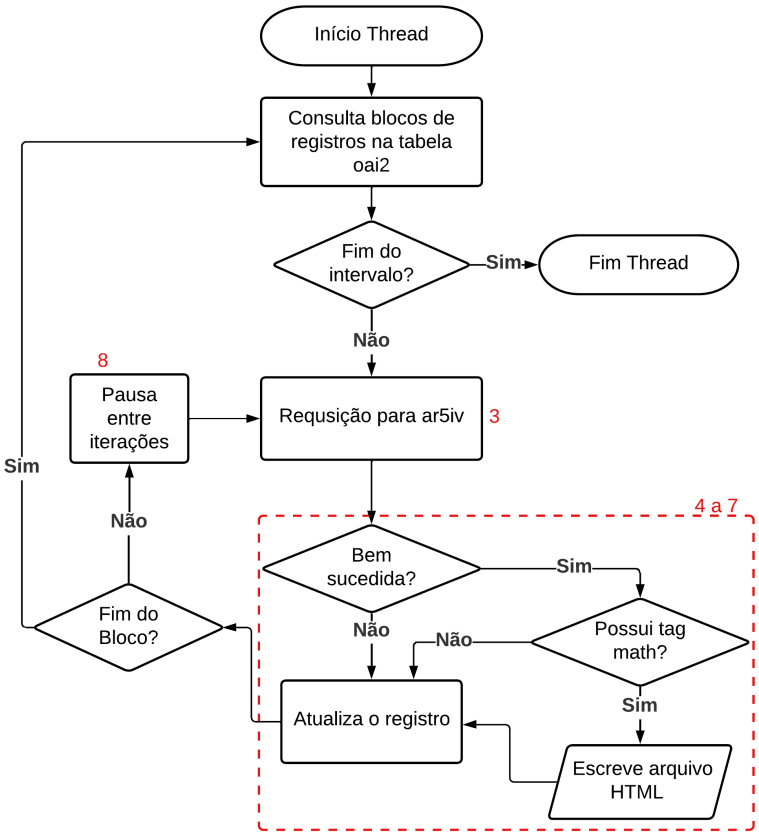

Figura 3. Lógica das *Threads* que iteram os intervalos.

4.3. Aplicação Java para Processamento

Após a execução da aplicação para *download* dos artigos convertidos para HTML5, obtidos no ar5iv, foi construída uma aplicação para processamento dos arquivos armazenados localmente. Sendo essa a segunda aplicação a compor este trabalho, desenvolvida também em Java, porém agora sem o auxílio de *Frameworks* como na primeira. Por ser uma aplicação pequena (em relação a quantidade de linhas de código), e que realiza tarefas críticas (execução em multithread, múltiplas conexões com banco de dados, e escrita massiva de arquivos), manipular todos esses componentes diretamente (sem o auxílio de *frameworks*), possibilitou a realização de ajustes mais finos, visando o máximo desempenho, e portanto, menor tempo de execução. Antes que se possa detalhar a aplicação responsável pelo processamento dos arquivos, faz-se necessário um entendimento básico da estrutura dos arquivos HTML5 obtidos à partir do ar5iv.

4.3.1. Estrutura do artigo no ar5iv (convertido em HTML5)

Os arquivos baixados pela aplicação anterior (em formato HTML5) possuem estrutura semelhante com a apresentada na Figura 4. O texto em si do artigo está representado na cor azul. Cada fórmula presente no artigo foi convertida para o formato MathML, resultando em *tags math*. Na figura é possível ver um *tag math* (destaca na cor vermelha), o que indica a presença de uma fórmula. Internamente à *tag* nota-se a fórmula em seu formato original (LaTeX), destacada na cor verde no atributo *alttext*. E por fim, a fórmula convertida para os formatos MathML *Presentation* e *Content*, respectivamente, se encontra destacada dentro dos retângulos na cor amarela.

Figura 4. Amostra da estrutura do arquivo HTML5.

Uma vez então entendida a estrutura do artigo em HTML5, a seguir é detalhada a saída a ser gerada pela aplicação.

4.3.2. Saída gerada pela aplicação

Ao se iniciar a análise de um artigo, as fórmulas são então identificadas, e o código LaTeX de cada uma delas é submetido a uma função *hash SHA-256*. A partir desse ponto, a aplicação gera dois tipos de saídas distintas:

O primeiro tipo, a ser persistido em um banco de dados *MySQL*, consiste no texto completo do artigo analisado, livre de *tags* HTML, contendo apenas *tags* especiais $<som>$ identificando as fórmulas. As *tags* do tipo $<som>$ são *tags* customizadas de elaboração própria para este projeto, e são utilizadas para identificar e substituir as *tags* de fórmulas nas páginas obtidas. Cada *tag som* terá um atributo *hash* (que identifica a fórmula), bem como o seu código LaTeX no corpo interno a *tag*.

O segundo tipo de saída são arquivos em formato MathML. Para cada fórmula existente no artigo, será gerado um arquivo cujo nome é o *hash* da fórmula acrescida da extensão *mml*, e o seu conteúdo, a fórmula em si escrita em formato MathML *Presentation*. A Figura 5 ilustra os passos aqui descritos.

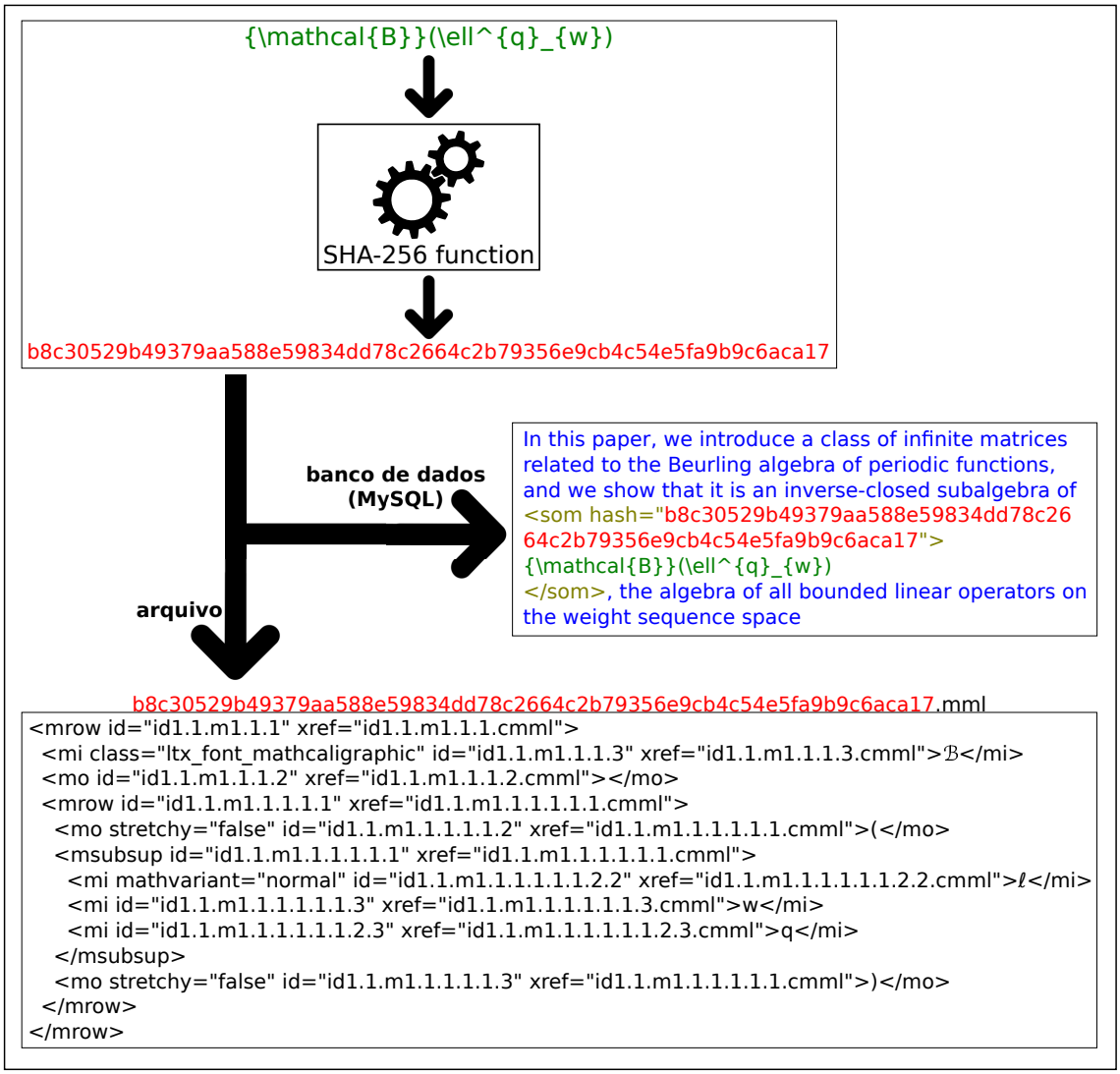

Figura 5. Saídas geradas pela aplicação.

É importante destacar o motivo de se gerar os arquivos de fórmulas separadamente do conteúdo persistido no Banco de Dados. Observe que, caso a ferramenta de busca, que irá indexar os dados produzidos por este trabalho, faça uso das fórmulas somente em formato LaTeX, os arquivos MathML *Presentation* podem ser desconsiderados, gerando-se assim uma quantidade menor de dados a serem indexados. No entanto, caso a ferramenta de busca precise das fórmulas em outro formato, por exemplo, por motivos de compatibilidade, poderá também fazer uso das fórmulas em formato MathML. Uma possível continuidade do trabalho aqui executado poderia ser, por exemplo, oferecer as fórmulas em outros formatos já renderizados, como: png, svg, dentre outros.

Uma vez então explicado o funcionamento da aplicação, a seção a seguir detalha o passo a passo de execução da mesma.

4.3.3. Execução da aplicação

A execução da aplicação segue portanto os passos descritos a seguir:

1. No momento em que a aplicação inicia, são definidos o número de *threads* para processamento, assim como tamanho dos blocos de registros a serem salvos na tabela *tb_content*;
2. Após a definição do número de *threads* e o tamanho dos blocos, o programa obtém a lista de arquivos HTML5 presentes localmente no servidor (baixados pela aplicação anterior);
3. Obtida a lista, é feita a divisão da mesma entre as *threads*, onde são definidos os intervalos de leitura para cada uma;
4. Definidos os intervalos a aplicação inicia o processamento de todas as *threads*;
5. Cada uma itera em seu intervalo de arquivos, de acordo com a seguinte lógica de execução:
   (a) O arquivo é lido do disco e é convertido para um objeto do tipo *Document*[3] da biblioteca *Jsoup*, objeto esse que encapsula o conteúdo de um HTML;
   (b) Através da biblioteca *Jsoup* é feita uma varredura no documento para mapeamento de todas as *tags math*;
   (c) Mapeadas as *tags*, uma a uma elas são iteradas para o processo de extração das fórmulas, que conta com os seguintes passos:
      i. Para cada fórmula obtém-se o atributo *alttext* da mesma, sendo este preenchido com a representação da fórmula no formato LaTeX;
      ii. A fórmula escrita em formato LaTeX passa então por uma função de *hash SHA-256*, algoritmo este utilizado nos demais sistemas do laboratório;
      iii. O *hash* gerado é utilizado como nome do arquivo que contém a fórmula em MathML;
      iv. O arquivo é então salvo em disco, contendo internamente a fórmula escrita em formato MathML *Presentation*;
      v. É considerada ainda a seguinte regra no momento da escrita do arquivo: o arquivo é gravado dentro de um diretório, cujo nome é composto pelas 3 primeiras letras do *hash* do nome do arquivo, evitando-se assim termos muitos arquivos em um único diretório;
      vi. Para evitar sobrescrita, sempre que for identificada a existência do arquivo, o mesmo não é salvo novamente e nem sobrescrito;
      vii. Uma vez que o documento da fórmula é salvo, sua *tag math* dentro do HTML é substituída por uma nova *tag* do tipo *som*, esta tem como atributo o campo *hash*, e seu valor é preenchido com o *hash* gerado na etapa 5(c)ii do fluxo de execução. O conteúdo interno da *tag* é a fórmula no formato LaTeX.
   (d) Feita a extração e substituição das fórmulas por completo, o conteúdo do arquivo passa pela remoção das demais *tags*, onde preservam-se apenas as *tags* do tipo *som* inseridas;
   (e) O conteúdo resultante do processo é agora submetido à persistência na base de dados, mais especificamente na tabela *tb_content*, que salva o conteúdo junto ao *id* do artigo.

[3] `https://jsoup.org/apidocs/org/jsoup/nodes/Document.html`

Os artigos são processados um a um pelo algoritmo acima, e no fim tem suas fórmulas mapeadas e salvas no servidor, assim como seu conteúdo pronto para um processo de indexação. Vale lembrar que todas as manipulações de elementos HTML no algoritmo são feitas com o auxílio do *Jsoup* e suas funcionalidades que fornecem um conjunto robusto de operações e opções para processamento de documentos HTML. Para acesso à base de dados e persistência dos conteúdos dos artigos pós processamento, fez-se necessário o uso do *Driver JDBC MySQL*

A Figura 6 ilustra a arquitetura descrita nesta seção.

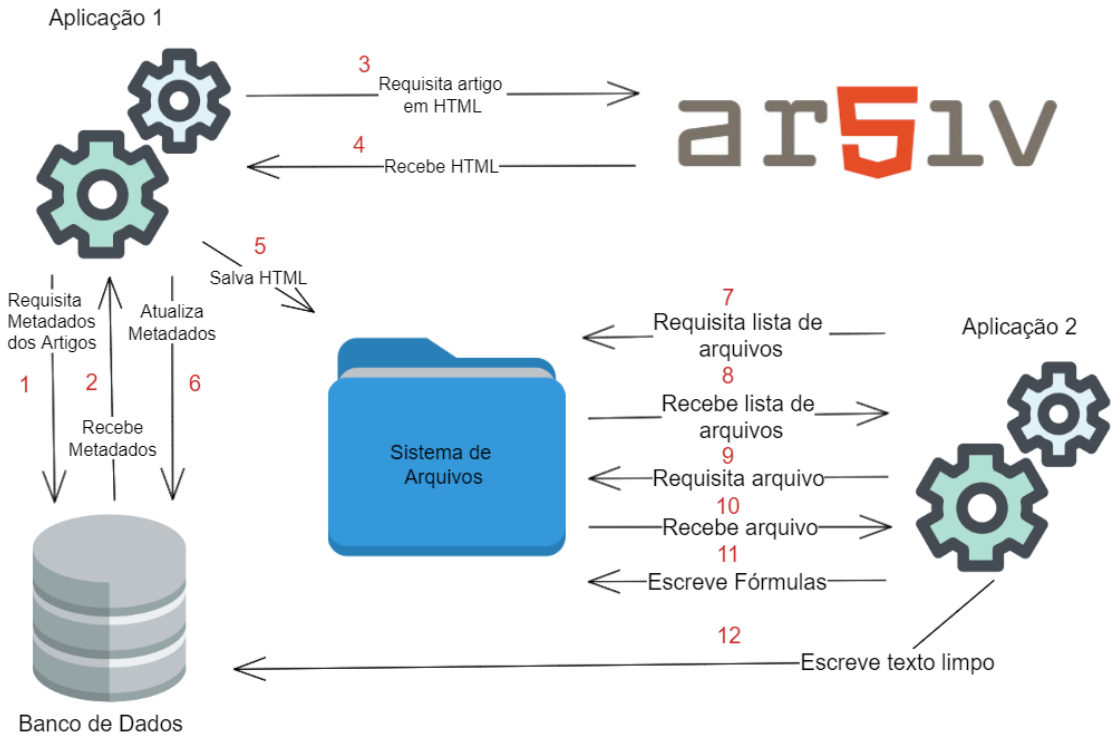

Figura 6. Arquitetura da Solução.

5. Resultados

Esta seção tem por objetivo apresentar os resultados de execução das duas aplicações propostas neste trabalho, assim como apontar observações feitas durante o processo.

Após a execução em 10 dias, a primeira aplicação obteve com sucesso um total de 1 829 120 de páginas HTML , valor que representa $88,4\%$ dos artigos presentes na base. Os demais artigos, ou não possuem fórmulas matemáticas em seu conteúdo, ou não foram convertidos para HTML até o momento da execução da aplicação, o que explica a diferença entre o número de páginas mapeado (2 068 457) e o número de fato obtido (1 829 120). A execução desta aplicação se deu em um longo prazo devido à necessidade de evitar a indisponibilidade do servidor da ar5iv, logo entre as requisições para *download* existiram pausas de 1 segundo, o que contribuiu para o longo prazo. A ocupação de disco gerada pelos artigos obtidos pela aplicação superou os 3 TB, ficando em torno dos $3,3$ TB divididos em dois SSDs que armazenaram respectivamente $1,6$ e $1,7$ TB cada.

Obtidas as páginas, deu-se início ao processamento da segunda aplicação. Para a execução desta etapa, foi alocada uma máquina do tipo *e2-standard-16* no ambiente *Google Cloud*. A máquina possui 16 núcleos e 64 GB de memória RAM. O tempo aproximado de execução foi de 1 dia. O tempo está aproximado porque durante a execução do programa, detectou-se um artigo em formato HTML5 que estava com problema na estrutura do seu código fonte (*1901.08676*[4]). Tal problema resultou na interrupção do algoritmo, e ações da equipe no sentido de isolar e reportar o problema[5]. Ao final de sua execução todas as 1 829 119 de páginas tiveram seus conteúdos textuais extraídos, ignorando todas as marcações de HTML, conservando-se apenas as marcações criadas para substituição da fórmula (este sub-processo foi apresentado na seção 4.3). No total foram extraídas e armazenadas 278 921 319 fórmulas matemáticas distintas. O espaço ocupado em disco para o armazenamento desta quantidade de fórmulas foi de 1, 6 TB, tendo sido necessário ainda aumentar a quantidade de *inodes* no disco utilizado, uma vez que o grande volume de arquivos fez com que a capacidade inicial de *inodes* do disco fosse alcançada. Assim, o disco foi esgotado inicialmente não por falta de espaço, mas por falta de entradas disponíveis na tabela de *inodes*.

Vale ressaltar que ambas as aplicações executaram seus processamento no formato *multithreading*, e contaram respectivamente com a parametrização de 6 e 12 *threads* durante suas execuções.

Porém, deixando de lado os problemas relatados, foi possível observar que ambas aplicações que compõem a solução para este trabalho desempenharam bem seus papéis, facilitando o processo de indexação da ar5iv, processo este a ser executado em um projeto futuro, utilizando-se das fórmulas e conteúdos provenientes dos artigos processados neste escopo.

6. Conclusão e Trabalhos Futuros

Este trabalho cumpriu com os objetivos propostos, como observado na seção 5, na qual foram apresentados o número de artigos obtidos, assim como o número de fórmulas extraídas. Logo é possível afirmar que ambas as aplicações desta solução cumpriram com os objetivos estabelecidos na seção 1, podendo portanto o trabalho desenvolvido servir como base para a indexação da ar5iv por ferramentas de busca de conteúdo matemático.

Uma recomendação é a limpeza das marcações geradas e adicionadas à *tag math* pelo ar5iv no processo de conversão dos artigos. A Figura 7 destaca em verde quais os segmentos a serem removidos no processo de limpeza sugerido. A figura é um exemplo de um conteúdo resultante da extração de uma *tag* feita neste projeto.

[4] `https://ar5iv.org/abs/1901.08676`
[5] `https://github.com/dginev/ar5iv/issues/326`

Figura 7. Segmentos a serem removidos em um processo de limpeza da *tag math*.

Um novo projeto futuro surge a partir deste trabalho, onde pode ser desenvolvida uma solução capaz de extrair não só o conteúdo MathML *Presentation* das fórmulas, como também o MathML *Content*. Como visto na Figura 4, as *tags math* presentes nas páginas dos artigos possuem em seu conteúdo o *Presentation* e o *Content*. Durante o processo de extração, o MathML *Content* é descartado, sendo extraído apenas o *Presentation*, como visto ao longo da seção 4.3. Uma futura abordagem pode levar em consideração a extração do MathML *Content* das fórmulas.

Referências

[ar5iv 2022] ar5iv (2022). arXiv articles as responsive web pages. https://blog.arxiv.org/2022/02/21/arxiv-articles-as-responsive-web-pages/ [Acessado em 13/02/2023].

[arXiv 1991] arXiv (1991). arXiv E-Print Archive. https://arxiv.org/ [Acessado em 13/02/2023].

[Baeza-Yates and Ribeiro-Neto 2013] Baeza-Yates, R. and Ribeiro-Neto, B. (2013). *Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca*. Bookman Editora.

[Codd 1970] Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387.

[Fielding et al. 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol–http/1.1. Technical report.

[Gierke et al. 2023] Gierke, O., Darimont, T., Strobl, C., Paluch, M., and Jay Bryant, Turnquist, G. (2023). Spring Data JPA–Reference Documentation. https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ [Acessado em 08/02/2023].

[Hedley 2009] Hedley, J. (2009). jsoup: Java HTML Parser. https://jsoup.org/ [Acessado em 08/02/2023].

[Hu et al. 2013] Hu, X., Gao, L., Lin, X., Tang, Z., Lin, X., and Baker, J. B. (2013). Wikimirs: a mathematical information retrieval system for wikipedia. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 11–20.

[Johnson et al. 2004] Johnson, R., Hoeller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T., Arendsen, A., Davison, D., Kopylenko, D., Pollack, M., et al. (2004). The Spring Framework–Reference Documentation. *interface*, 21:27.

[Manning 2008] Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing.

[MariaDB Corporation 2023] MariaDB Corporation (2023). *MariaDB 10.5 Documentation*. `https://mariadb.com/docs/` [Acessado em 06/02/2023].

[Mozilla Contributors 2023a] Mozilla Contributors (2023a). Web/HTML. `https://developer.mozilla.org/pt-BR/docs/Web/HTML` [Acessado em 08/02/2023].

[Mozilla Contributors 2023b] Mozilla Contributors (2023b). Web/MathML. `https://developer.mozilla.org/pt-BR/docs/Web/MathML` [Acessado em 13/02/2023].

[MySQL 2023] MySQL (2023). *MySQL 8.0 Reference Manual*. `https://dev.mysql.com/doc/refman/8.0/en/` [Acessado em 06/02/2023].

[Oracle Corporation 2023] Oracle Corporation (2023). JDBC Technology. `https://docs.oracle.com/en/database/oracle/oracle-database/19/lnjbj/index.html` [Acessado em 07/02/2023].

[Rodrigues and Gonzaga 2019] Rodrigues, B. d. O. and Gonzaga, F. B. (2019). Obtenção de fórmulas matemáticas da base de artigos arxiv.

[Sojka 2012] Sojka, P. (2012). Exploiting semantic annotations in math information retrieval. In *Proceedings of the fifth workshop on Exploiting semantic annotations in information retrieval*, pages 15–16.

[The Spring Team 2023] The Spring Team (2023). Spring Cloud OpenFeign. `https://docs.spring.io/spring-cloud-openfeign/docs/current/reference/html/` [Acessado em 08/02/2023].

[Webb et al. 2023] Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C., Deleuze, S., Simons, M., Pavić, V., Bryant, J., Bhave, M., Meléndez, E., Frederick, S., and Halbritter, M. (2023). Spring Data Boot–Reference Documentation. `https://docs.spring.io/spring-boot/docs/3.0.2/reference/htmlsingle/` [Acessado em 08/02/2023].