UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Rafael Tabarin Redondo

ALGORITMOS PARA A EXTRAÇÃO DE FÓRMULAS DE BASES DE DADOS DO *STACK EXCHANGE* PARA A FERRAMENTA *SEARCHONMATH*

Alfenas, 07 de Julho de 2015.

UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ALGORITMOS PARA A EXTRAÇÃO DE FÓRMULAS DE BASES DE DADOS DO *STACK EXCHANGE* PARA A FERRAMENTA *SEARCHONMATH*

Rafael Tabarin Redondo

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Alfenas como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Flávio Barbieri Gonzaga

Alfenas, 07 de Julho de 2015.

Rafael Tabarin Redondo

ALGORITMOS PARA A EXTRAÇÃO DE FÓRMULAS DE BASES DE DADOS DO *STACK EXCHANGE* PARA A FERRAMENTA *SEARCHONMATH*

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Prof. Leonardo Aparecido Ciscon

Universidade Federal de Alfenas

Profa. Mariane Moreira de Souza

Universidade Federal de Alfenas

Prof. Flávio Barbieri Gonzaga

Universidade Federal de Alfenas

Alfenas, 07 de Julho de 2015.

RESUMO

A *SearchOnMath* é uma ferramenta que realiza busca por fórmulas matemáticas em diversas bases de dados existentes na *Web*, como *Wikipedia* e *Mathworld*. Com a necessidade de expansão e crescimento da ferramenta, foi proposta para este trabalho a inclusão em sua base de dados de fórmulas existentes em fóruns hospedados no site *Stack Exchange*. Os fóruns identificados com potencial foram o *Mathematica*, o *Mathematics* e o *MathOverFlow*. As bases de dados dos 3 fóruns foram obtidas através de arquivos *d*isponibilizados pelo *Stack Exchange*. Foram desenvolvidos algoritmos para importar as páginas de cada fórum e extrair suas fórmulas, inserindo em uma base de dados com a mesma estrutura da base de dados da ferramenta SearchOnMath. A extração das fórmulas é um problema bastante difícil porque nem sempre a marcação da fórmula dentro do texto da página é feita de maneira correta, além da existência de comandos especiais e de falta de padronização da linguagem utilizada para construção das fórmulas. Isso faz com que trechos de texto sejam reconhecidos de maneira equivocada, como se fossem fórmulas. Este problema é detalhado e tratado nos algoritmos desenvolvidos. Como resultado final, foi obtido um número considerável de novas fórmulas, o que resultará futuramente no aumento da capacidade de busca da ferramenta.

Palavras-Chave: *SearchOnMath*, *Stack Exchange*, Busca de Fórmulas Matemáticas, Extração de Fórmulas Matemáticas.

ABSTRACT

SearchOnMath is a tool that performs search for mathematical formulas in several databases existing in the web, like Wikipedia and Mathworld. With the need of expansion and growth of the tool, the inclusion of existing formulas in Stack Exchange hosted forums was proposed for this work. The identified forums with potential were Mathematica, Mathematics and MathOverFlow. The databases of the three forums were obtained from files provided by Stack Exchange. Algorithms were developed to import the each forum pages and extract your formulas, inserting in a database with the same structure of the database of SearchOnMath tool. The formula extraction is a very difficult problem because not always the formula labelling inside the text is done correctly, in addition to the existence of special commands and lack of standardization of the language used to build the formulas. This makes text pieces are recognized wrongly, as formulas. This problem is detailed and treated in the developed algorithms. As a final result, was obtained a considerable number of new formulas that will result in future an increase of the tool search capability.

Keywords: SearchOnMath, Stack Exchange, Retrieval of Mathematical Formulas, Extraction of Mathematical Formulas.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

CSV	Comma-Separed Values
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
SQL	Script Query Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

SUMÁRIO

1
Introdução

Este capítulo descreve o cenário das ferramentas de busca na rede e, especificamente, de busca por fórmulas matemáticas antes do surgimento da ferramenta SearchOnMath, explica o funcionamento básico da mesma e expõe as justificativas, a problematização e os objetivos gerais e específicos deste trabalho.

Com o contínuo crescimento da Internet, bem como dos recursos disponíveis, surgem a cada dia novos desafios que vão desde a recuperação de conteúdo de interesse de cada usuário bem como da exibição dos recursos de forma agradável, onde pode-se citar como exemplo o *Google*, que permite visualizar uma miniatura das páginas retornadas, sem que seja necessário acessá-las. Ferramentas de busca são uma alternativa na recuperação desse conteúdo, que justamente pelo crescimento da rede mundial, necessitam de algoritmos escaláveis, e de mecanismos que facilitem buscas específicas. Ao se analisar a evolução da quantidade de consultas feitas por dia em ferramentas de busca, observa-se a crescente demanda por esses serviços: em 1994, a *World Wide Web Worm* recebia cerca de 1500 consultas por dia; em 1997, o Altavista realizava cerca de 20 milhões de consultas por dia (Brin, 1998). A estimativa em 2009 era de que só nos Estados Unidos, algo próximo a 300 milhões de consultas eram realizadas por dia no *Google*, desconsiderando o uso de demais ferramentas como o Bing e o *Yahoo!* (Oreskovic et al., 2009).

Além do formato tradicional das ferramentas citadas, que buscam a partir de palavras, as empresas começam a voltar o foco para buscas mais inteligentes e de conteúdo específico. Alguns exemplos são a busca por artigos científicos, realizada pelo *Scholar*[1] do *Google*; além da ferramenta de conhecimento computacional *Wolfram Alpha*[2], onde o usuário pode obter relatórios elaborados de forma automática com base na consulta realizada.

Ao se observar a tendência no desenvolvimento de ferramentas capazes de realizar buscas específicas, pode-se listar alguns conteúdos não tratados pelas ferramentas acima

xixxix

[1] http://scholar.google.com/

[2] http://www.wolframalpha.com/

citadas, e que começam a receber a atenção. Um dos exemplos é a busca por fórmulas matemáticas.

Propostas de trabalhos para a busca matemática começaram a surgir por volta do ano de 2006 (Kohlhase et al., 2006, Shatnawi et al., 2007, Asperti et al., 2006), e vem se intensificando nessa década (Avny et al., 2006, Xiaoyan et al., 2014, Kamali et al., 2013, Xuan et al., 2013).

A proposta do presente trabalho se concentra na área da busca matemática. Justificativa e Motivação

Com base no funcionamento da ferramenta *SearchOnMath*, observa-se que a mesma oferecia a busca por fórmulas matemáticas em 5 bibliotecas. Visando a expansão da ferramenta, mostrou-se necessário oferecer suporte a mais bibliotecas.

Outro fator que se mostra importante, é a padronização de alguns módulos do sistema. A ferramenta possui muitas etapas manuais na atualização de sua base de pesquisa e na inserção de novos sites e bibliotecas na mesma. Com a padronização, essas etapas podem ser realizadas de forma mais automática.

1.1 Problematização

Para a expansão da ferramenta *SearchOnMath*, neste trabalho foram escolhidos os fóruns *Mathematica* e *Mathematics* como novos dados a serem incluídos na base de pesquisa, e a atualização dos dados do fórum *MathOverFlow*. Os 3 fóruns são hospedados no site *Stack Exchange*, que disponibiliza a base de todos os seus fóruns para download. A base de dados de cada fórum é disponibilizada em um arquivo zip, que contém vários arquivos *xml*, divididos por tipos de dados, como *Comments*, *Posts* e *Users*. Essa estrutura de dados deve ser interpretada e importada para a base de dados da ferramenta, que contém apenas os dados necessários para o seu funcionamento. Após a importação dos dados, é necessário extrair as fórmulas existentes nos dados, executar os algoritmos de tratamento e de interpretação léxica de cada fórmula extraída já existentes na ferramenta e analisar os resultados, identificando erros cometidos pelos usuários para analisar se convém tratá-los. Todo esse processo deve ser feito de forma mais automática e padronizada possível, evitando processos e algoritmos para bases de dados específicas.

1.2 Objetivos

1.2.1 Gerais

Inserir as bases de dados *Mathematica* e *Mathematics* na ferramenta *SearchOnMath* e atualizar a base de dados *MathOverFlow* na mesma.

1.2.2 Específicos

- Estudo avançado de Programação;
- Estudo avançado de Banco de Dados;
- Estudo da ocorrência de fórmulas nos fóruns *Stack Exchange*;
- Download de arquivos das bases de dados *Mathematica*, *Mathematics* e *MathOverFlow*;
- Importação desses arquivos em um Banco de Dados com a estrutura da base de dados da ferramenta *SearchOnMath*;
- Extração de fórmulas das bases de dados importadas.

1.3 Organização da Monografia

Este trabalho está organizado em 8 capítulos.

O presente capítulo (Introdução) descreve o cenário das ferramentas de busca na rede e, especificamente, de busca por fórmulas matemáticas antes do surgimento da ferramenta *SearchOnMath*, explica o funcionamento básico da mesma e expõe as justificativas, a problematização e os objetivos gerais e específicos deste trabalho.

O capítulo 2 (Revisão Bibliográfica) descreve trabalhos já existentes na área de busca por fórmulas matemáticas na rede e introduz o contexto em que o trabalho foi baseado.

O capítulo 3 (Referencial Teórico) apresenta o embasamento teórico para o entendimento deste trabalho.

O capítulo 4 (Metodologia) introduz a disposição das tabelas da base de dados da ferramenta SearchOnMath e mostra os algoritmos desenvolvidos para atingir os objetivos do trabalho.

O capítulo 5 (Resultados) apresenta os resultados obtidos após o desenvolvimento e execução dos algoritmos aplicando-os nas bases de dados trabalhadas.

O capítulo 6 (Conclusões e Trabalhos Futuros) contém as considerações finais da monografia, considerando os objetivos e os resultados obtidos, bem como propostas para desenvolvimento de trabalhos futuros.

O capítulo 7 (Referências) relaciona as referências bibliográficas desta monografia.

O capítulo 8 (Anexos) apresenta os anexos inseridos nesta monografia.

2
Revisão Bibliográfica

Este capítulo descreve trabalhos já existentes na área de busca por fórmulas matemáticas na rede e introduz o contexto em que o trabalho foi baseado.

2.1 Cenário da Busca Matemática

O desenvolvimento de ferramentas de busca de fórmulas matemáticas na rede é recente. Os trabalhos relacionados começaram a ser apresentados em 2006 e vêm evoluindo com o tempo. Os trabalhos mais recentes apresentados são os das ferramentas *WikiMirs* e *Tangent*, que trabalham com a biblioteca *Wikipedia*, além da ferramenta *SearchOnMath*.

2.2 WikiMirs

O *WikiMirs* é uma ferramenta criada para facilitar a busca por fórmulas matemáticas no *Wikipedia*, proposta em "*A Mathematics Retrieval System for Formulas in Layout Presentations*" (Lin et al., 2013). A ferramenta realiza a busca por fórmulas matemáticas similares baseada tanto nas similaridades textuais como espaciais, utilizando um modelo de indexação e comparação desenvolvida em estrutura de camadas. No modelo, uma técnica de generalização hierárquica é utilizada para gerar sub-árvores das árvores que representam a fórmula matemática, e a similaridade é calculada de acordo com o nível de correspondência nos diferentes níveis dessas árvores.

2.3 Tangent

O motor de busca *Tangent*, proposto em "*Math Expression Retrieval Using an Inverted Index Over Symbol Pairs*" por Stalnaker e Zanibbi (2015), é um sistema que utiliza um índice invertido sobre pares de símbolos de expressões matemáticas, onde cada índice é um par de símbolos com sua distância relativa e deslocamento vertical dentro de uma expressão. As expressões são ranqueadas pela média da porcentagem de pares de símbolos coincidentes na expressão buscada, e pela média da porcentagem de pares de símbolos coincidentes na expressão candidata.

2.4 Base de Dados *Wikipedia*

Tanto o *WikiMirs* como o *Tangent* utilizam como base de dados o *Wikipedia* para realizar a busca de fórmulas, que disponibiliza a sua base de dados para *download*. As páginas vem em um arquivo *XML*, onde as expressões matemáticas são delimitadas por *tags* <math> e representadas em *LaTeX*. No caso do *Tangent*, essas expressões são extraídas das páginas *HTML* juntamente com seus artigos associados e convertidas para a representação canônica *MathML2* usando *LaTeXML*. No *WikiMirs* as fórmulas são extraídas e mantidas em *LaTeX*.

2.5 O problema no contexto do *Stack Exchange*

No contexto dos fóruns hospedados no *Stack Exchange* (www.stackexchange.com), as fórmulas são inseridas de forma diferente, seguindo o padrão da linguagem *TeX*. Apesar de parecer semelhante à estrutura de fórmulas do *Wikipedia*, o padrão da linguagem *TeX* se torna mais complicado para a extração de fórmulas, e será explicado no Capítulo 3, bem como toda a estrutura do *Stack Exchange*. Este trabalho foi desenvolvido neste contexto e trata os desafios contidos no mesmo.

3
Referencial Teórico

Este capítulo apresenta o embasamento teórico para o entendimento deste trabalho.

3.1 SearchOnMath

Em constante desenvolvimento no LaReS (Laboratório de Redes de computadores e Sistemas distribuídos), a ferramenta *SearchOnMath*[3] (Gonzaga, 2013) surgiu como uma alternativa bastante eficiente na busca por fórmulas matemáticas. A ferramenta foi criada oferecendo a busca por fórmulas matemáticas nas seguintes bibliotecas:

- *Wikipedia* (porção matemática da biblioteca em inglês)[4];
- *MathWorld*[5];
- DLMF (*Digital Library of Mathematical Functions*)[6].

Atualmente a *SearchOnMath* oferece suporte a mais dois sites:

- *PlanetMath*[7];
- *MathOverFlow*[8].

Desde a ideia inicial até o estado atual do projeto já se passaram cerca de 7 anos. Quando o seu desenvolvimento foi iniciado, não se encontrava à disposição na Internet ferramentas semelhantes. Era comum encontrar artigos sobre o assunto, mas que não tinham gerado um produto em si (ou o produto já não se encontrava mais online). Assim, o desenvolvimento inicial da *SearchOnMath* foi bastante desafiador em função da falta de

xxvxxv

[3] http://searchonmath.com/

[4] http://en.wikipedia.org/

[5] http://mathworld.wolfram.com/

[6] http://dlmf.nist.gov/

[7] http://planetmath.org/

[8] http://mathoverflow.net/

exemplos práticos. E muito do código-fonte desenvolvido precisou ser adaptado depois, à medida que a ferramenta cresceu.

No estado atual de desenvolvimento, o projeto *SearchOnMath* é bastante complexo, e possui basicamente os módulos ilustrados na Figura 1.

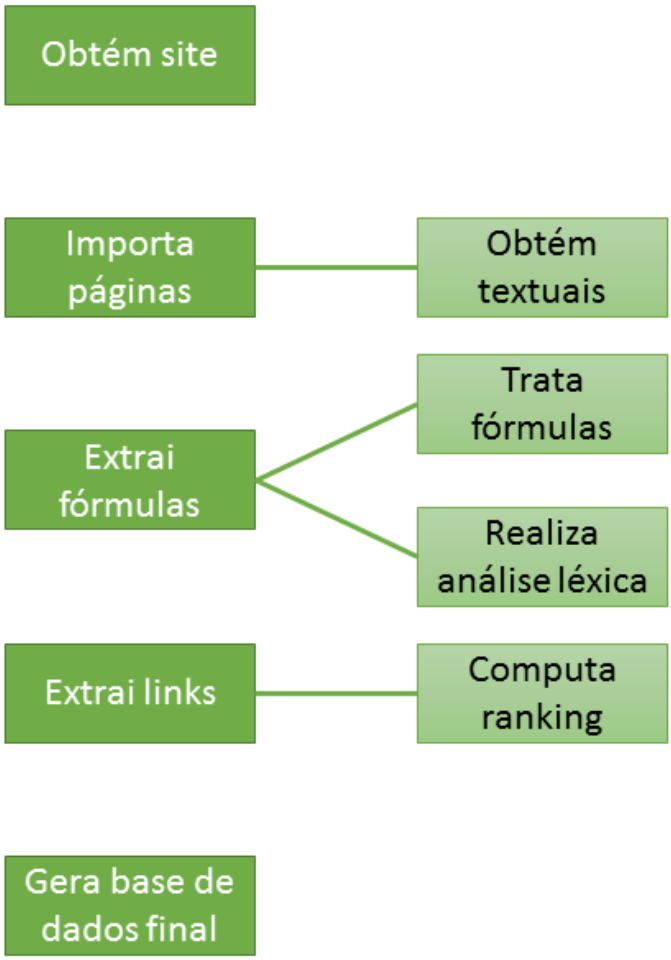

Figura 1 - Estrutura básica dos módulos da *SearchOnMath*.

- "Obtém site": composto internamente por um *WebCrawler* (feito em *Python*). Alguns sites disponibilizam a base de dados pronta, caso esse que dispensa o uso do *WebCrawler* (caso da *Wikipedia* e do *MathOverFlow*).
- "Importa páginas": percorre os arquivos obtidos pelo módulo anterior, e realiza a importação para o banco dados. Internamente ele possui uma subdivisão, chamada "Obtém textuais". Essa subdivisão é a responsável por extrair das páginas elementos que serão exibidos na busca depois, como título e resumo. Os códigos são desenvolvidos em Java.

- "Extrai fórmulas": realiza a extração das fórmulas analisando as páginas importadas para o banco de dados no passo anterior. Internamente possui duas subdivisões: "Trata fórmulas" e "Realiza análise léxica". A "Trata fórmulas" realiza ações como remover elementos textuais das fórmulas (vírgulas, pontos... quando ocorrem no início ou no final da fórmula, e indica que é um elemento que pertence ao texto da página); substituir *tags HTML* pelos respectivos comandos em *LaTeX*. A "Realiza análise léxica" extrai os *tokens* das fórmulas, que alimentarão a *SearchOnMath* depois. Esses módulos são desenvolvidos em *Java* e C.
- "Extrai *links*": Extrai os *links* das páginas e importa para uma tabela no banco de dados. Internamente tem a subdivisão "Computa *ranking*", que a partir dos links, computa diversas medidas de ranqueamento, que depois auxiliam na ordenação dos resultados. Esse módulo é desenvolvido em *Java*.
- "Gera base de dados final": Percorre todas as tabelas geradas nas etapas anteriores, e gera um novo banco de dados bastante enxuto, só com as informações necessárias ao funcionamento da *SearchOnMath*. Esse é o banco de dados que alimenta a ferramenta no final. Módulo também feito em *Java*.

Além dos módulos descritos acima, existem ainda uma série de detalhes, bem como a *SearchOnMath* em si, que é desenvolvida em JSF (*Java Server Faces*).

3.2 *Stack Exchange*

O *Stack Exchange* é uma rede de mais de 130 comunidades de perguntas e respostas (fóruns). Começando com o fórum *Stack Overflow*, destinado a programadores e interessados em programação, hoje hospeda diversos outros fóruns de áreas diversas, como matemática, melhorias domésticas, estatística, língua inglesa, entre outras, onde os usuários podem procurar, perguntar e responder a respeito desses assuntos.

Todos os fóruns da rede possuem administradores, classificação de usuários e de postagens, onde usuários comuns podem ser eleitos administradores, de acordo com suas qualificações obtidas com participações nos fóruns. As postagens também podem ser classificadas através de votos positivos ou negativos dos usuários e receber marcações de

favoritos e de melhor resposta, pois podem ser postagens de pergunta ou postagens de resposta.

3.2.1 Obtenção da base de dados

As bases de dados do *Stack Exchange* podem ser obtidas através do *Dump* de cada fórum, disponibilizado pelo próprio *Stack Exchange*, no endereço https://archive.org/download/stackexchange.

3.2.1.1 *Dump*

Um *Dump* é um registro da base de dados, normalmente utilizado para backup do sistema em questão, em caso de perda de dados. Contém uma cópia da base de dados atualizado até a data da criação do mesmo. Também pode ser utilizado para reuso de dados, e normalmente são publicados por software livre e projetos de conteúdo livre, como os fóruns do *Stack Exchange*.

3.2.2 A estrutura

3.2.2.1 Base de dados

A base de dados obtida através do *Dump* tem uma estrutura de tabelas própria, que é descrita no Anexo I.

3.2.2.2 A ocorrência de fórmulas

No *Stack Exchange*, a ocorrência de fórmulas é identificada pelos caracteres '$', que funcionam como marcadores de início e fim de uma fórmula. Os marcadores '$' são utilizados para iniciar e finalizar um trecho de fórmula no texto. Os marcadores '$$' iniciam e finalizam um bloco de fórmula, ou seja, ficam 'separados' do texto, centralizados e em linhas separadas. Na Figura 2, os dois casos de ocorrência de fórmulas são exemplificados e a Tabela 1 mostra o trecho da postagem em que as fórmulas ocorrem.

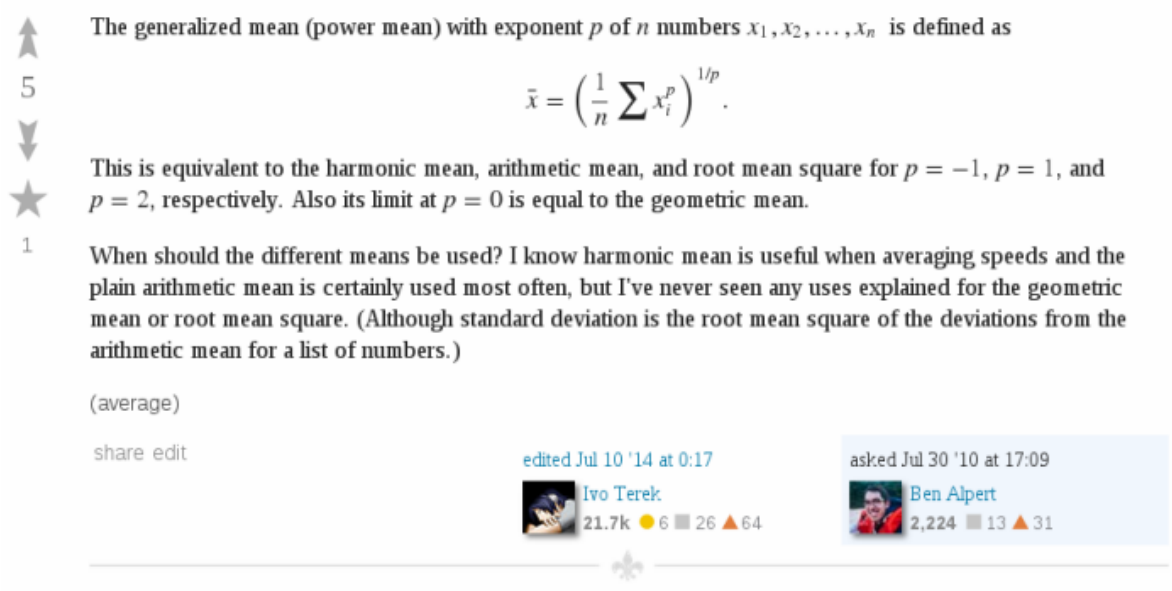

Figura 2 - Exemplo de postagem com ocorrência de fórmulas simples e em bloco

Tabela 1 - Exemplo destacado de ocorrências de fórmulas simples e em bloco

The generalized mean (power mean) with exponent **p** of **n** numbers **$x_1, x_2, \ldots, x_n$** is defined as

 $$ \bar x = \left(\frac{1}{n} \sum x_i^p\right)^{1/p}. $$

 This is equivalent to the harmonic mean, arithmetic mean, and root mean square for **$p = -1$**, **$p = 1$**, and **$p = 2$**, respectively. Also its limit at **$p = 0$** is equal to the geometric mean.

 When should the different means be used? I know harmonic mean is useful when averaging speeds and the plain arithmetic mean is certainly used most often, but I've never seen any uses explained for the geometric mean or root mean square. (Although standard deviation is the root mean square of the deviations from the arithmetic mean for a list of numbers.)

Como pode ser observado na Tabela 1 e na Figura 2, o trecho de fórmula que é identificado com o marcador '$$' é renderizado em um bloco separado do texto, mesmo estando na mesma linha e no meio do texto. Já no trecho de fórmula identificado pelo marcador '$' a fórmula é renderizada na mesma linha do texto.

3.2.3 Bases de dados trabalhadas

Neste trabalho, as bases de dados utilizadas para expansão da base de dados da ferramenta *SearchOnMath*, foram as dos fóruns *Mathematica*, *Mathematics* e *MathOverFlow*.

3.2.3.1 *Mathematica*

O fórum do *Mathematica* no *Stack Exchange* (http://mathematica.stackexchange.com/) é dedicado a usuários do programa *Wolfram Mathematica*, definido como um sistema definitivo para computação técnica moderna (http://www.wolfram.com/mathematica/). A sua base de dados contém 57393 registros de postagens, das quais apenas 8220 apresentam fórmulas em seu interior.

3.2.3.2 *Mathematics*

O fórum *Mathematics* no Stack *Exchange* (http://math.stackexchange.com/) é dedicado a pessoas que estudam matemática em todos os níveis de aprendizado e profissionais da área. Nele, os usuários podem fazer perguntas relacionadas a matemática, bem como responder e procurar por postagens de seu interesse. A sua base de dados contém 1002727 postagens, das quais 865241 apresentam fórmulas em seu interior.

3.2.3.3 *MathOverFlow*

O fórum *MathOverFlow* (http://mathoverflow.net/) é dedicado a matemáticos profissionais. Da mesma forma, os usuários podem procurar, postar e responder postagens relacionadas à área matemática. A sua base de dados contém 156407 postagens, das quais 109485 apresentam fórmulas em seu interior.

3.3 Problemas na extração de fórmulas

3.3.1 Elementos Textuais

Na linguagem *TeX* é possível inserir trechos de texto dentro de um trecho de fórmula, através de comandos de elementos textuais, como os comandos \text, \mbox, \fbox, \raisebox, \hbox, etc. Um exemplo de uma postagem com um trecho de texto dentro de uma fórmula pode ser visto na Figura 3, e seu respectivo trecho de código *HTML* na Tabela 2.

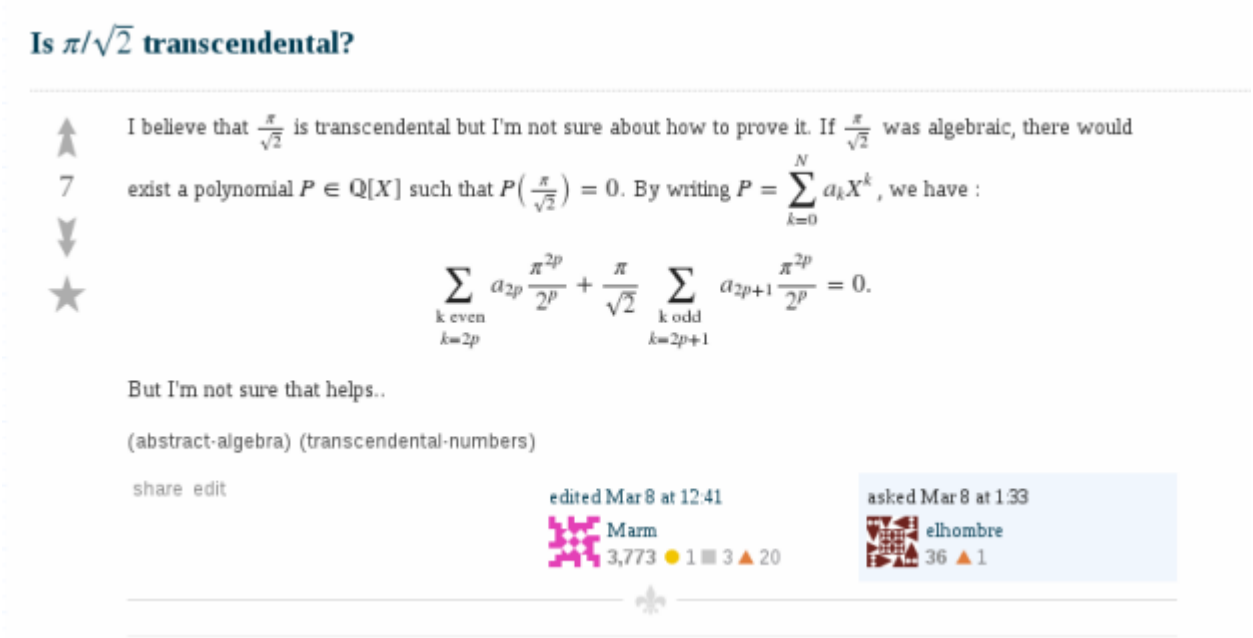

Figura 3 - Exemplo de postagem com o uso de um comando de elemento textual

Nota-se que o parâmetro de dois somatórios existentes na postagem são textos, o 'k even' e o 'k odd'. Portanto, usa-se o comando \text para conseguir inserir esses trechos de texto dentro de elementos da fórmula, como pode ser visto no trecho de código destacado na Tabela 2.

Tabela 2 - Trecho de código da postagem exibida na figura 3, com os elementos textuais destacados.

I believe that $\frac{\pi}{\sqrt{2}}$ is transcendental but I'm not sure about how to prove it. If $\frac{\pi}{\sqrt{2}}$ was algebraic, there would exist a polynomial $P \in \mathbb{Q}[X]$ such that $P\big(\frac{\pi}{\sqrt{2}} \big) = 0$. By writing $P = \displaystyle \sum_{k=0}^{N} a_{k}X^{k}$, we have :

 $$ \sum_{\substack{**\text{k even}** \\[1mm] k = 2p}} a_{2p} \frac{\pi^{2p}}{2^{p}} + \frac{\pi}{\sqrt{2}} \sum_{\substack{**\text{k odd}** \\[1mm] k = 2p+1}} a_{2p+1} \frac{\pi^{2p}}{2^p} = 0. $$

 But I'm not sure that helps..
 |

3.3.1.1 O problema

O problema enfrentado neste trabalho relacionado a esses elementos textuais das fórmulas *TeX* se dá pelo fato de que na linguagem é possível inserir uma fórmula dentro de um elemento textual. Ou seja, se dentro de um trecho iniciado pelo comando \text, o usuário inserir um

caractere ‘$’, o compilador lê como um trecho de fórmula. Resumindo, a linguagem aceita a inserção de fórmulas dentro de elementos textuais que estão dentro de uma outra fórmula. Sendo assim, o usuário pode utilizar deste recurso, ao invés de encerrar o trecho de elemento textual, inserir a fórmula, e iniciar outro trecho de elemento textual. Essa abertura da linguagem a diversas formas de escrever um mesmo elemento dentro de uma fórmula é o maior desafio enfrentado neste trabalho. Portanto, na extração das fórmulas da postagem, deve-se considerar que um caractere ‘$’ dentro de um elemento textual está iniciando outro trecho de fórmula, e não encerrando o anterior. Da mesma forma, a fórmula inserida dentro do elemento textual deve ser extraída como uma fórmula a parte. Para melhor entendimento do problema, um exemplo dessa ocorrência pode ser visto abaixo na Tabela 3. Novamente os comando de elemento textual estão destacados e, as fórmulas dentro do mesmo estão sublinhadas.

Tabela 3 - Trecho de código da postagem exibida na Figura 4, com os elementos textuais destacados.

I am a little confused about the weak* topology on Hilbert space H. Beyond doubt, the weak* topology on H^{**} is $\sigma(H^{**},H^*)$. Suppose τ is the natural embedding from H onto H^{**}. Then a topology $\mathcal{T_{w*}}$ on H induced by τ is linear isomorphism and homeomorphism to $\sigma(H^{**},H^*)$. In fact, $\mathcal{T_{w*}}=\{ U \subseteq H : \tau(U) ~ **\text{is an open set in $\sigma(H^{**},H^*)$} \}**$. Hence, we may regard $(H,\mathcal{T}_{w^*})$ as the desired weak* topology on H. Moreover, $(H,\mathcal{T_{w*}})$ is also a locally convex space. Since H is a reflexive space, we should have $(H, \mathcal{T_{w*}})=(H,\sigma(H,H^*))$, that is $\mathcal{T_{w*}} = \sigma(H,H^*)$. For the definition of weak* topology on H is not exactly the same as that of usual case, hence, I have used the word "should" which means that I am not so sure about it.

 If this equation $\mathcal{T_{w*}} = \sigma(H,H^*)$ is true, can anyone tell me in detail why? Thanks!
 |

Na Figura 4 pode-se observar o exemplo da Tabela 3 renderizado, com a ocorrência de uma fórmula dentro de um trecho de texto inserido em uma fórmula destacada:

Weak* topology on Hilbert space

I am a little confused about the weak:* topology on Hilbert space H. Beyond doubt, the weak:* topology on H^{**} is $\sigma(H^{**}, H^{*})$. Suppose τ is the natural embedding from H onto H^{**}. Then a topology $\mathcal{T}_{w*}$ on H induced by τ is linear isomorphism and homeomorphism to $\sigma(H^{**}, H^{*})$. In fact, $\mathcal{T}_{w*} = \{U \subseteq H : \tau(U)$ is an open set in $\sigma(H^{**}, H^{*})\}$. Hence, we may regard $(H, \mathcal{T}_{w*})$ as the desired weak:* topology on H. Moreover, $(H, \mathcal{T}_{w*})$ is also a locally convex space. Since H is a reflexive space, we should have $(H, \mathcal{T}_{w*}) = (H, \sigma(H, H^{*}))$, that is $\mathcal{T}_{w*} = \sigma(H, H^{*})$. For the definition of weak:* topology on H is not exactly the same as that of usual case, hence, I have used the word "should" which means that I am not so sure about it.

If this equation $\mathcal{T}_{w*} = \sigma(H, H^{*})$ is true, can anyone tell me in detail why? Thanks!

(functional-analysis) (hilbert-spaces) (topological-vector-spaces) (locally-convex-spaces) (reflection)

share edit

edited Mar 7 at 15:38

asked Mar 7 at 14:36
Travis Wang
381 ■ 2 ▲ 10

add a comment

Figura 4 – Exemplo de postagem com ocorrência de fórmula dentro de um trecho textual inserido em uma fórmula

Sendo assim, o algoritmo deve tratar esses casos específicos em que pode existir uma fórmula identificada por '$' dentro de um trecho de fórmula em bloco (identificado por '$$') ou de fórmula simples (identificado por '$'). O algoritmo não considera trechos de fórmula em bloco dentro de um trecho de fórmula simples, pois a linguagem não permite esta abordagem.

No caso exemplificado na Tabela 3 e ilustrado na Figura 4, se o algoritmo somente procurar por um fechamento pelo caractere '$', seria extraído erroneamente o seguinte trecho: $\mathcal{T_{w*}}=\{ U \subseteq H : \tau(U) ~ **\text{is an open set in $**, quando na verdade deveria extrair os trechos de fórmula: $\mathcal{T_{w*}}=\{ U \subseteq H : \tau(U) ~ **\text{is an open set in $\sigma(H^{**},H^{*})$} \}$** e **$\sigma(H^{**},H^{*})$**. Nota-se que a fórmula inserida no comando de elemento textual, deve ser extraído separadamente, mas também como parte da fórmula a qual o mesmo está inserido.

3.3.2 Modificadores de Estilo

Da mesma forma que existem os comandos que indicam elementos de texto, existem os comando que modificam o estilo do trecho da fórmula, como os comandos \mathbb, \mathrm e \mathop. Estes comandos trabalham apenas com trechos de fórmula, ou seja, elementos textuais só podem ser inseridos nos mesmos se tiverem dentro de um comando de elemento textual. Portanto, não podem ser inseridos trechos de fórmula dentro destes comandos, mas

alguns usuários cometem este equívoco. Sendo assim, um estudo foi necessário para decidir se esses comandos deveriam ser tratados de forma exclusiva ou não. Como fórmulas não podem ser inseridas, não é necessário tratá-los. Porém, com os erros dos usuários nesses casos, algumas fórmulas são extraídas de forma errônea, já que um caractere '$' dentro de um modificador de estilo encerra a fórmula, e compromete a extração das fórmulas seguintes dentro da postagem. Na Figura 5 um exemplo da utilização do modificador de estilo é ilustrado. Como pode ser observado, na primeira linha existem duas fórmulas onde a letra 'Z' tem o seu estilo modificado.

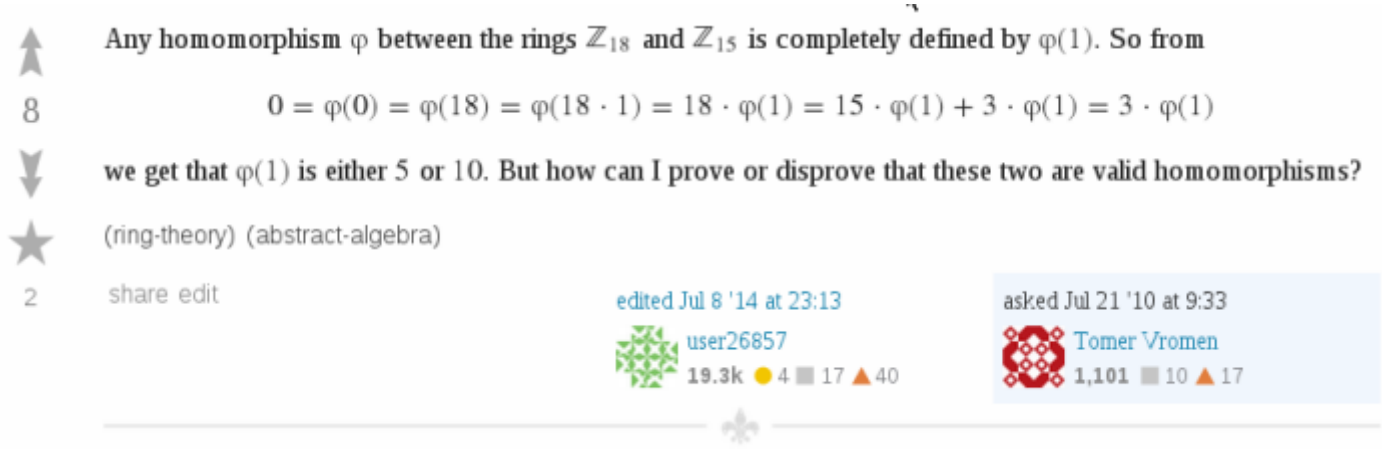

Figura 5 - Exemplo de postagem com modificadores de estilo

Na Tabela 4 abaixo, existe um exemplo de ocorrência de identificador de fórmula dentro de um comando modificador de estilo, o que geraria um erro na extração das fórmulas.

Tabela 4 - Exemplo de ocorrência de identificador de trecho de fórmula dentro de um comando modificador de estilo

Here's the problem:

 <img src="http://i.stack.imgur.com/Mm4oW.jpg" alt="The problem">

 Now, here is what I did:

<blockquote>
 Since we invested $\mathbb{$}1000$ at the beginning of $1989,1990$, and 1991, we find the balance of each contribution over the specified intervals of time and sum them together. So, denoting C_{y} to be the year in which our contribution was made, we obtain:
 $$C_{1989}=1000(1.06)(1.055)(1.05)(1.045)\approx 1227.05$$
 $$C_{1990}=1000(1.065)(1.06)(1.055)(1.05)\approx 1250.54$$
 $$C_{1991}=1000(1.06)(1.0555)(1.05)(1.05)\approx 1232.93$$
 $$\implies B= C_{1989}+C_{1990}+C_{1991}=3710.52$$
 So our balance in 1994 is $B=\mathbb{$}3710.52$.
</blockquote>

 Apparently, my answer is off by 3 dollars, as the answer is claimed to be 3713.16.To my question: was my approach to the problem incorrect? There wasn't much to go off of in the chapter, so I wasn't quite sure how to play with this one other than one example.
 |

No caso do exemplo da Tabela 4, será extraído o trecho $B=\mathbb{$. Se o usuário se atentasse ao problema e inserido um caractere de escape para o '$', seria extraído corretamente o trecho $B=\mathbb{\$}3710.52$.

3.3.3 O caractere '$'

Como o caractere '$' indica um início ou término de trecho de fórmula no *TeX*, alguns usuários comentem o erro de inseri-los sem a intenção de iniciar ou terminar uma fórmula. Esse equívoco gera, da mesma forma que o caso dos Modificadores de Estilo, uma extração errônea de fórmula, extraindo trechos do código *HTML* e inserindo-os na base de dados da ferramenta como fórmula. Nestes casos, o correto é a utilização do caractere de escape '\' antes do caractere '$'. Assim, o compilador da linguagem vai reconhece-lo como um caractere de texto, e não um identificador de trecho de fórmula.

3.3.4 Escapes do caractere '$'

Com a existência dos escapes para o caractere '$', o algoritmo deve reconhecer que o mesmo, se antecipado pelo escape, não deve ser tratado como identificador de fórmula. Sendo assim, deve ignorá-lo, e reconhecê-lo como um caractere comum.

3.3.5 Escapes do caractere ‘\’

Da mesma forma que no caso anterior, existe o escape para o caractere ‘\’. Ou seja, se o usuário desejar inserir o caractere ‘\’ como um caractere comum, e não como escape, deve “escapá-lo”. Sendo assim, o algoritmo também deve reconhecê-lo como um caractere comum, caso seja antecipado pelo caractere de escape. Exemplificando, no trecho $x= \text{\$}5.25$, o caractere ‘$’ destacado não será reconhecido como fechamento de fórmula, logo o caractere de fechamento da fórmula estará mais adiante. Já no trecho $$r(3u_1) + r(2) - r(u_3) = 0 \\$, o caractere ‘\’ está precedido por um outro caractere ‘\’, fazendo com que se torne um caractere comum, e não de escape. Sendo assim, o caractere ‘$’ seguinte será reconhecido como fechamento de fórmula.

3.3.6 Comentários em fórmulas

Na linguagem *TeX*, existe a possibilidade de inserção de comentários nos trechos de fórmulas. O início de um comentário é identificado pelo caractere ‘%’, e só termina quando o compilador encontra uma quebra de linha (‘\n’). Ou seja, um trecho de comentário é iniciado por um ‘%’ e encerrado por um ‘\n’. Portanto, todos os trechos de comentário devem ser desconsiderados da fórmula, pois não podem ser inseridos como tal na base de dados da ferramenta, já que são trechos textuais.

3.3.7 Escapes do caractere ‘%’

Assim como nos dois casos de escape anteriores, o algoritmo deve entender que um caractere ‘%’ precedido por ‘\’ é na verdade um caractere comum, e não um início de trecho de comentário dentro da fórmula.

4
Metodologia

Este capítulo introduz a disposição das tabelas da base de dados da ferramenta SearchOnMath e mostra os algoritmos desenvolvidos para atingir os objetivos do trabalho.

4.1 Estrutura das tabelas da base de dados da *SearchOnMath*

A ferramenta *SearchOnMath* possui uma base de dados onde as fórmulas originais e tratadas e as páginas de onde foram extraídas são armazenadas. A consulta de fórmulas quando uma busca é realizada é feita nesta base de dados, que possui 4 tabelas: *tb_equation*, *tb_html*, *rl_html_equation* e *tb_link*.

4.1.1 Tabela *tb_equation*

A tabela *tb_equation* armazena as fórmulas que foram extraídas das páginas e postagens. Os campos que compõem a mesma são:

- *equ_id* : identificador da tupla;
- *equ_equation*: fórmula original;
- *equ_equation_ttd*: fórmula tratada (após passar pelo algoritmo tratador da ferramenta, que trata códigos *HTML* e outros caracteres existentes na fórmula original);
- *equ_equation_lex*: fórmula transformada em expressão léxica (após passar pelo algoritmo léxico);

4.1.2 Tabela *tb_html*

A tabela *tb_html* armazena as páginas e postagens originais que foram importadas das bases de dados das bibliotecas e fóruns. Os campos existentes na mesma são:

- *htm_id*: identificador da tupla;
- *htm_url*: *url* da página/postagem extraída;
- *htm_title*: título da página/postagem;
- *htm_src*: conteúdo *HTML* da página/postagem;
- *htm_abstract*: resumo da página/postagem/
- *htm_rank*: índice ranking gerado de acordo com o nível de importância da página/postagem, utilizado para ordenação de retorno de consultas;

4.1.3 Tabela *rl_html_equation*

A tabela *rl_html_equation* armazena as relações entre equações e as páginas/postagens onde ocorrem. É uma tabela que gera uma relação muitos para muitos entre as tabelas *tb_equation* e *tb_html*, ou seja, uma tupla da primeira pode estar relacionada com mais de uma tupla da tabela *rl_html_equation* e uma *tupla* da segunda pode estar relacionada com mais de uma tupla da tabela *rl_html_equation*. Os campos desta tabela são:

- *htm_id*: identificador da tupla da tabela de páginas/postagens, chave estrangeira da tabela *tb_html*;
- *equ_id*: identificador da tupla da tabela de equações, chave estrangeira da tabela *tb_equation*;

4.1.4 Tabela *tb_link*

A tabela *tb_link* armazena os links que cada página/postagem contém e é utilizada para gerar um índice *rank* para cada página/postagem. No entanto, neste trabalho o índice *rank* é gerado a partir do pontuação (campo score da tabela posts das bases de dados *Stack Exchange*) de cada postagem. Sendo assim, a tabela *tb_link* não foi utilizada e seus campos não serão especificados neste trabalho.

4.2 Algoritmos desenvolvidos

O presente trabalho foi desenvolvido em 4 etapas:

- Obtenção das bases de dados *Stack Exchange* e importação das mesmas para uma base de dados *MySQL*;
- Importação das bases *MySQL* do *Stack Exchange* para a base de dados da ferramenta *SearchOnMath*;
- Extração das fórmulas contidas nas postagens;
- Geração de gráfico e dados para análise dos resultados.

Cada uma das etapas compreende o desenvolvimento ou adaptação de um algoritmo e testes do mesmo.

É importante destacar que cada um dos três fóruns é trabalhado separadamente, ou seja, as quatro etapas são executadas para a base de dados de um dos fóruns, para só depois executá-las para a base de dados do próximo fórum. Além disso, inicialmente, cada um dos 3 fóruns possui uma base de dados com a estrutura da ferramenta *SearchOnMath*, para, no final das etapas das três bases, juntá-las com a base de dados única da *SearchOnMath*. Essa última parte não é feita neste trabalho. A Figura 6 ilustra o cenário das bases de dados trabalhadas e seu fluxo de transformações. Neste capítulo, as bases de dados de cada fórum que possuem a estrutura da *SearchOnMath* serão tratadas como base de dados da *SearchOnMath*, já que a base final não é utilizada neste trabalho. As etapas são descritas e detalhadas nos subcapítulos a seguir.

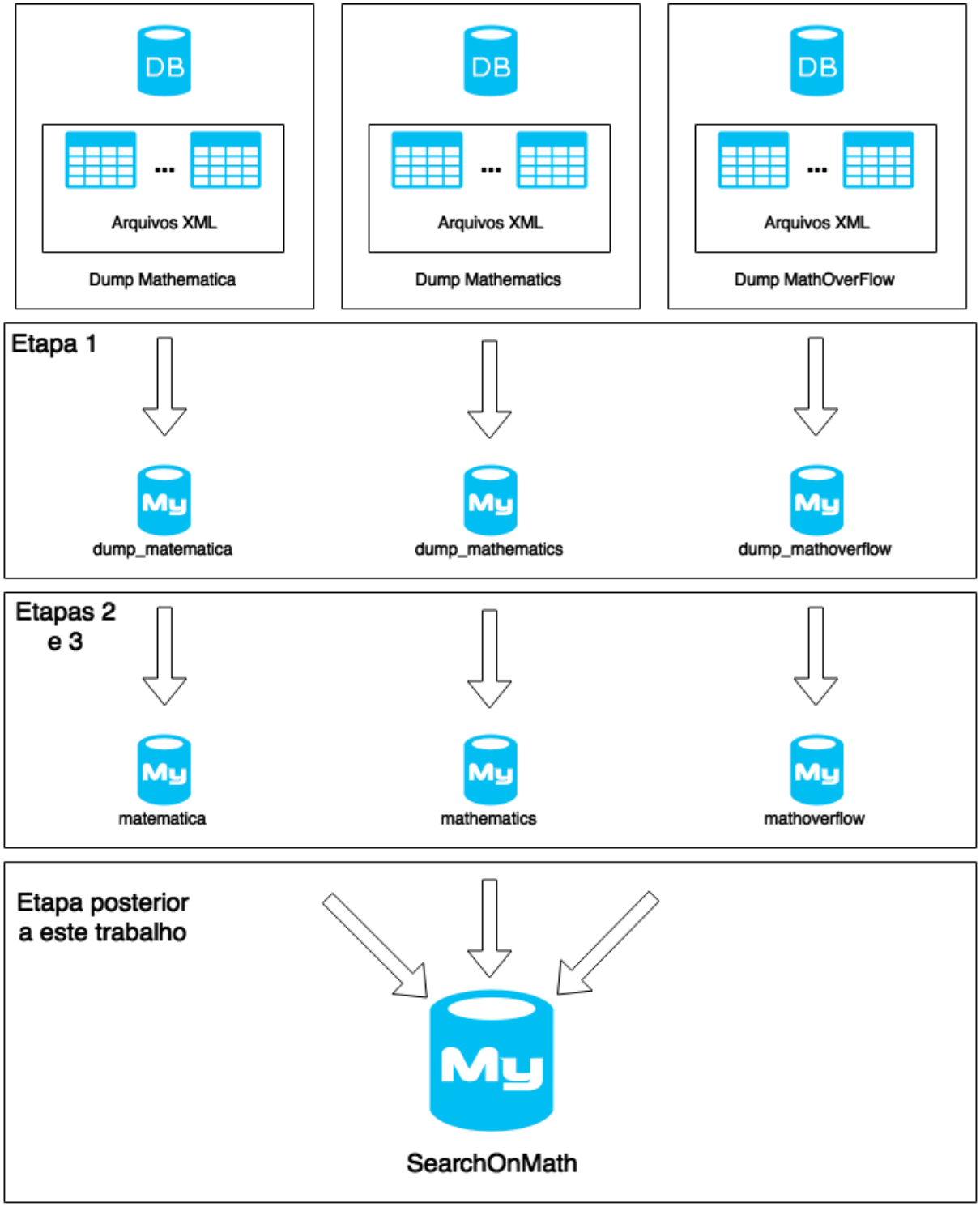

Figura 6 - Cenário das bases de dados trabalhadas e seu fluxo de transformações

4.2.1 Etapa 1: Obtenção das bases de dados *Stack Exchange* e importação das mesmas para uma base de dados *MySQL*

As bases de dados do *Stack Exchange* são disponibilizadas para download em arquivos *XML*. As versões das bases trabalhadas são de 09 de Março de 2015 e foram obtidas em 27 de Março de 2015, através do repositório disponibilizado pelo *Stack Exchange*. Cada arquivo *XML* corresponde a uma tabela dessa base de dados. A importação dos dados dessas tabelas foram baseadas em um *script SQL* encontrado após pesquisa na *Web*. Esse script foi postado pelo usuário do *GitHub* megansquire (https://github.com/megansquire) e está disponível em https://gist.github.com/megansquire/877e028504c92e94192d. O *script* é bem simples e contém a criação da base de dados *MySQL*, a criação das tabelas *MySQL* correspondentes a cada arquivo *XML*, o carregamento dos arquivos *XML* em cada uma das tabelas criadas e a definição de índices para os campos das mesmas. O *script* foi adaptado para gerar uma base de dados para cada um dos três fóruns trabalhados.

Após a geração das bases de dados *MySQL*, foram realizados testes para verificar se os dados importados correspondem com o que está acessível nos fóruns na *web*. Como exemplo, na Figura 7 observa-se uma postagem do fórum *Mathematics* com sua versão na base de dados *MySQL*.

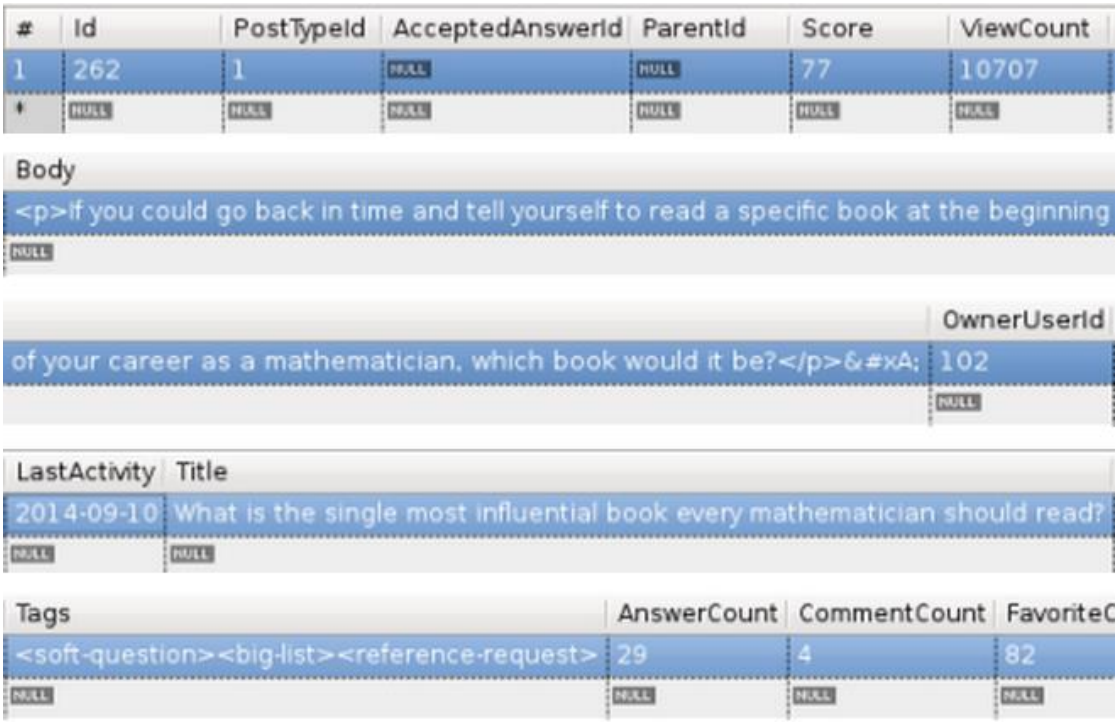

Figura 7 - Exemplo de postagem do fórum *Mathematics* na base de dados *MySQL*, visualizada no *MySQL Workbench*

A Figura 8 apresenta a mesma postagem mostrada na Figura 7 renderizada na página web do fórum. Pode-se observar que o título, o conteúdo, as *tags* e a data de última edição são idênticas ao que está na base de dados *MySQL*. O *Score* é diferente (81 contra 77) pois a data de criação do *Dump* é anterior à data da visualização da postagem na página *web*, portanto, já passou por atualizações. Essa postagem pode ser acessada em http://math.stackexchange.com/questions/262/.

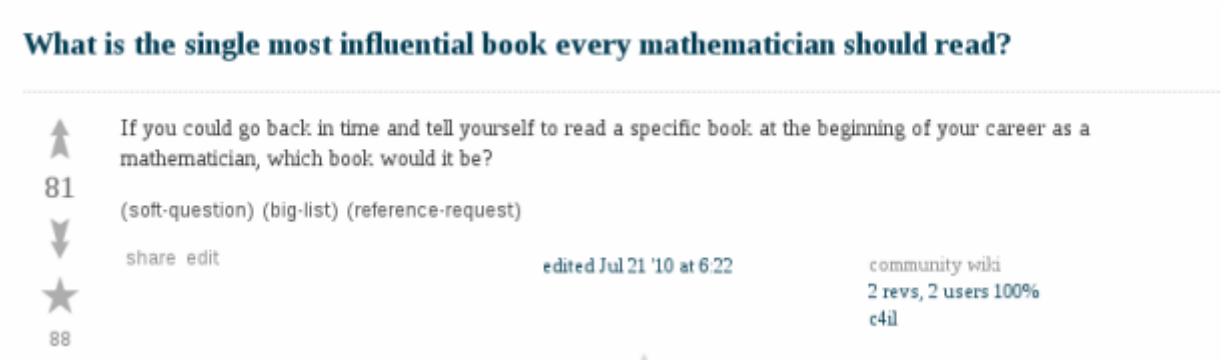

Figura 8 - Exemplo de postagem do fórum *Mathematics* na página *web* do fórum

4.2.2 Etapa 2: Importação das bases *MySQL* do *Stack Exchange* para a base de dados da ferramenta *SearchOnMath*

Após a importação dos arquivos *XML* para a base de dados *MySQL*, é necessário importar essas bases para a base de dados da ferramenta *SearchOnMath*. Nesta etapa, apenas a tabela *tb_html* é populada, para posteriormente, na etapa 3, as fórmulas serem extraídas a partir da mesma.

Foi desenvolvido um algoritmo que lê as tuplas da tabela *posts* da base de dados *MySQL*, as interpreta e as insere na tabela *tb_html*.

Inicialmente, o algoritmo realiza uma consulta *MySQL* que retorna o *score* mínimo e a soma total do *score* entre todas as postagens do fórum. Isso é feito para normalizar os valores de *rank* entre 0 e 1, já que o *score* foi definido como sendo o valor de *rank* das postagens dos fóruns do *Stack Exchange*. O cálculo desse valor normalizado se dá pelo valor do *score* dividido pela soma total dos *scores* de todas as postagens. Essa normalização é feita em todas as importações de bases de dados da ferramenta *SearchOnMath*. Como existem *scores* de postagens que têm valor negativo, é necessário buscar o menor valor de *score* para somá-lo a todos os demais, fazendo com que todas as postagens tenham valor de *rank* maior que ou igual a 0. Essa soma é realizada antes do processo de normalização.

Depois do armazenamento dos valores de *score* (mínimo e total) nas variáveis *minScore* e *totalScores*, o algoritmo busca os dados da tabela de postagens. Um comando SQL busca os campos *id*, *PostTypeId*, *Title*, *Body*, *Tags* e *Score* de cada tupla da tabela *posts*. Os campos *id, Title*, *Body* e *Tags* são inseridos nos campos *htm_id*, *htm_title*, *htm_src* e *htm_abstract* da tabela *tb_html* sem nenhum tipo de tratamento ou alteração. Notem que as *tags* da postagem são armazenadas como o resumo da postagem, pois assim serão exibidas na tela de resultados da ferramenta *SearchOnMath*, para as bases de dados *Stack Exchange*. Já o campo *PostTypeId* é utilizado para identificar o tipo de postagem e assim, construir a *URL* da página *web* correspondente. A *URL* das postagens do *Stack Exchange* tem a seguinte estrutura: "http://" + nome do fórum + ".stackexchange.com/questions/" + *id* da postagem. Um exemplo da postagem com *id* número 208112 pode ser visto na Tabela 5.

Tabela 5 - Exemplo de URL construída para o fórum *MathOverFlow*

http://mathoverflow.stackexchange.com/questions/208112/

As *URL's* construídas pelo algoritmo e armazenadas na base de dados são, quando executadas no navegador, redirecionadas para o padrão de cada fórum que, normalmente, contêm o título da postagem ao final da *URL*. Na Figura 9 observa-se a *URL* exemplificada na Tabela 5 após o redirecionamento para o padrão do fórum *MathOverFlow*.

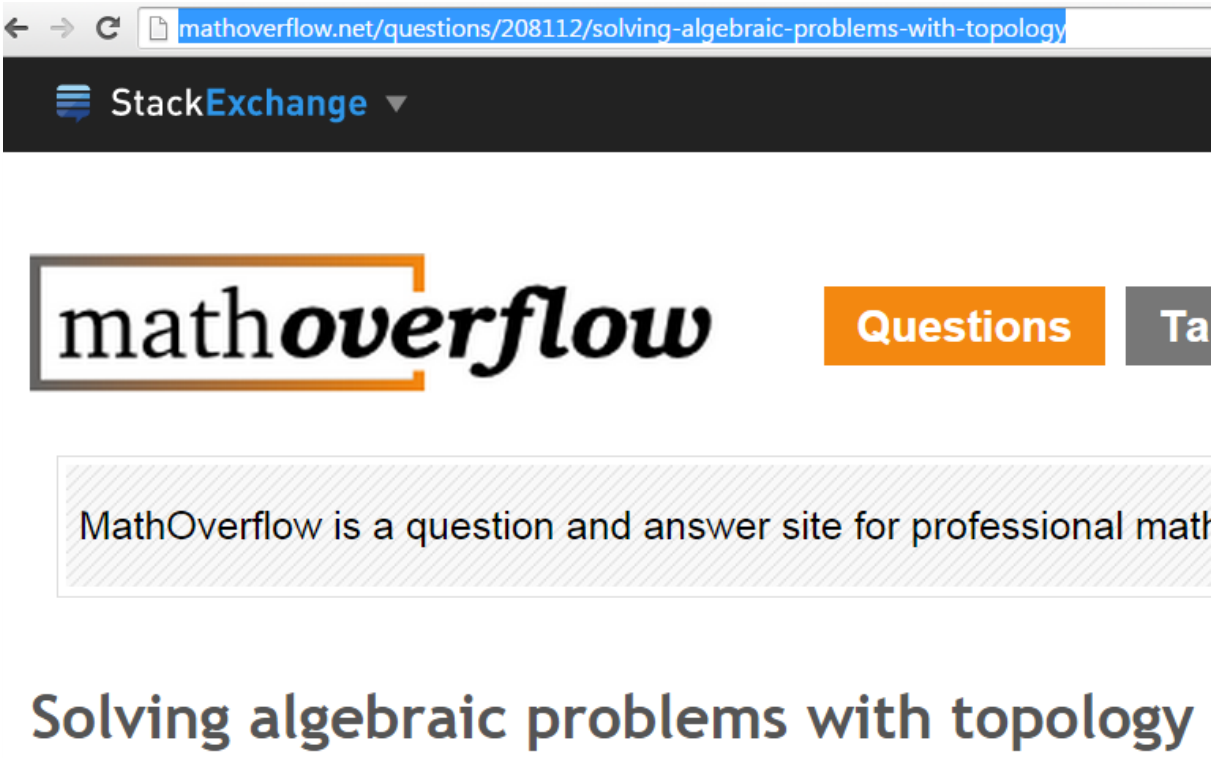

Figura 9 - Exemplo de *URL* de postagem no *MathOverFlow*

No entanto, esta estrutura só é válida para as postagens que têm o *PostTypeId* com valores 1, 2 ou 6. Para as postagens com os valores 4 e 5, indicando postagem relacionada à uma página de *Tag*, a estrutura é definida como "http://" + nome do fórum + ".stackexchange.com/tags/" + nome da *tag* + "/info". O nome da *tag* é recuperado da tabela *tags* na base de dados *MySQL* importada do *Dump* do *Stack Exchange*, em um comando *SQL* que tem a condição de que o campo *ExcerptPostId* ou o campo *WikiPostId* sejam iguais ao *id* da postagem na tabela *posts*. A Tabela 6 contém um exemplo de *URL* de uma postagem do tipo *tag* construída pelo algoritmo.

Tabela 6 - Exemplo de *URL* de postagem do tipo *Tag* construída pelo algoritmo

http://mathoverflow.stackexchange.com/tags/reference-request/info

Da mesma forma, a *URL* das postagens do tipo *Tag* podem ter sua *URL* redirecionada para uma *URL* padronizada com o fórum, como pode ser visto na Figura 10, para o mesmo caso da *URL* da Tabela 6.

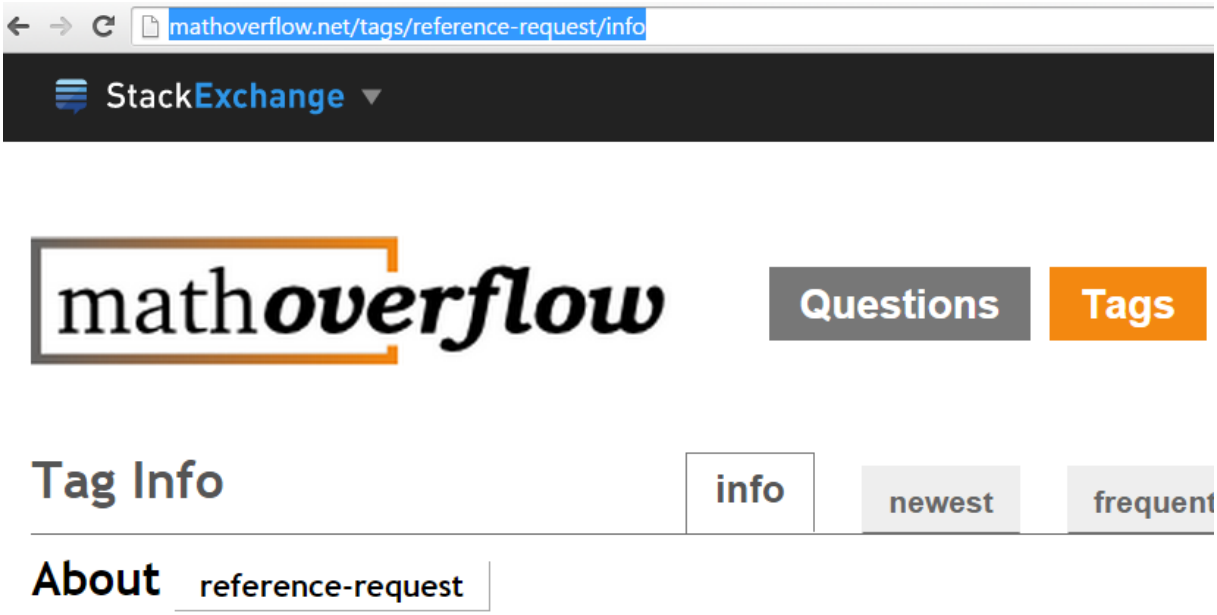

Figura 10 - Exemplo de *URL* de postagem do tipo *Tag* no *MathOverFlow*

A *URL* construída pelo algoritmo é armazenada no campo *htm_url* da tabela *tb_html* da base de dados da ferramenta *SearchOnMath*. As postagens do tipo *Tag* que não tinham o seu id relacionado com nenhuma tupla da tabela *rank* foram desconsideradas, por ser impossível construir uma *URL* correspondente.

Por fim, o campo *score* é utilizado para o cálculo do valor de *rank*, somando-o com o valor da variável *minScore* (para eliminar valores negativos). O resultado dessa soma é divido pelo valor da variável *totalScores*, fazendo com que o *rank* esteja sempre entre 0 e 1. O *rank* é armazenado no campo *htm_rank* da tabela *tb_html*.

O algoritmo executa estes procedimentos para todas as tuplas da tabela *posts*.

Os testes desta etapa foram feitos comparando os campos de algumas tuplas com as postagens originais na *web*, verificando se os títulos, corpo e *tags* das postagens são os mesmos da base de dados.

4.2.3 Extração das fórmulas contidas nas postagens

Nesta etapa, os campos *htm_src* e *htm_title* da tabela *tb_html* são lidos para a procura de fórmulas em seu interior. A primeira consulta só retorna os campos que contém pelo menos dois caracteres '$', indicando a possibilidade de ocorrência de fórmulas. Os casos em que isso não acontece são desconsiderados, pois indicam erro do usuário ou que o caractere é utilizando como texto, já que uma fórmula deve ser aberta e encerrada por pelo menos um caractere '$'.

O conteúdo dos campos *htm_src* e *htm_title* são tratados da mesma forma no algoritmo, já que podem conter fórmulas da mesma maneira e por isso serão chamados apenas de conteúdo neste de capítulo.

Neste algoritmo, a identificação de trechos de fórmula ocorre basicamente por meio da separação do conteúdo em um vetor. Essa separação é feita por meio do comando *Java split*, aplicado em *Strings*, que são divididas em um vetor a partir de uma expressão regular utilizada como separador. Na Figura 11 é mostrado um exemplo de um conteúdo antes e depois da separação em vetor através do *split*, com o separador definido como '$$'.

- String antes do *split:*

```
<p>For example: $$\int_0^1(15-x)^2(\text{d$2$}x)^2$$</p>&#xA;
```

- Vetor após o *split*:

0	For example:
1	\int_0^1(15-x)^2(\text{d2}x)^2
2

Figura 11 - Exemplo de conteúdo antes e depois do *split* com o separador definido como '$$'

Como as fórmulas podem ser identificadas por '$' e por '$$', os dois casos devem ser tratados. Porém, um trecho de fórmula simples ('$') não pode conter um trecho de fórmula em bloco ('$$'), mas o contrário pode acontecer. Assim, primeiramente é feito o *split* com o separador definido como '$$'.

Porém, antes do *split*, é adicionado um caractere branco (' ') ao final do conteúdo para que, em caso de término do conteúdo com um '$$' encerrando a fórmula, adicione mais um índice ao final do vetor quando executar o *split*. O *split* gera um vetor onde todas os índices ímpares indicam trecho de fórmula, já que, mesmo que o conteúdo inicie com um '$$', o split coloca um valor vazio no primeiro índice do vetor e, portanto, o primeiro separador, que indica o início de um trecho de fórmula, é o que vai separar os índices 0 e 1 do vetor e, o segundo separador, que indica o término do trecho de fórmula, é o que vai separar os índices 1 e 2 do vetor, definindo o índice 1 como um trecho de fórmula, como pode ser observado nas Figura 11. O mesmo ocorre para os índices seguintes do vetor e, indica que todos os índices ímpares do vetor são trechos de fórmula. Porém, se um índice ímpar do vetor for também o último, ele é desconsiderado, pois indica um trecho de fórmula que foi iniciado mas não encerrado, por erro de usuário, já que em caso de o último trecho do conteúdo ser uma fórmula, um índice em branco foi adicionada ao final do vetor, fazendo com que o último índice seja par.

Após a separação, as *Strings* dos índices ímpares são armazenadas em um *ArrayList* chamado *equacoes*. O *ArrayList* é um vetor de objetos da classe Equação, que é formada por uma *String equacao* e um *int htm*_id, que armazenam a fórmula a ser inserida na base de dados e o *id* da postagem à qual ela pertence.

Em seguida, o processo todo é repetido, mas a separação através do *split* é feita para cada um dos índices do primeiro vetor gerado e com o separador definido com '$'. Essa nova separação é feita dentro de todos os índices do primeiro vetor gerado pois um trecho de fórmula simples pode ocorrer dentro ou fora de um trecho de fórmula em bloco. Na Figura 12, observa-se o segundo *split* feito, já utilizando o separador definido como '$'. No caso, o *split* é executado no conteúdo do índice 1 do vetor gerado no exemplo da Figura 11.

- String antes do *split:*

```
\int_0^1(15-x)^2(\text{d$2$}x)^2
```

- Vetor após o *split*:

0	\int_0^1(15-x)^2(\text{d
1	2
2	}x)^2

Figura 12 - Exemplo de conteúdo antes e depois do *split* com o separador definido como '$'

É importante ressaltar que os separadores '$' e '$$' não podem ser precedidos pelo caractere '\', que indicaria o escape, tornando o caractere '$' um elemento textual, e não um identificador de trecho de fórmula. Essa restrição é definida na expressão regular que parametriza os *splits*. Ou seja, o conteúdo não será dividido no ponto em que existir um '$' ou um '$$' precedido por um '\'. Além disso, os caracteres '\' também não podem ser precedidos por um caractere de escape '\', pois também se tornariam elementos textuais ao invés de escape, e os '$' e '$$' seriam novamente indicadores de trecho de fórmula. Para resolver este problema, antes de qualquer um dos *splits*, todos as ocorrências de '\\' foram substituídas por '\\ ' (com um espaço no final), evitando que o segundo caractere '\' funcione como escape para um '$', já que é na verdade um elemento textual.

Outro problema que acontece nesta etapa é que, como a expressão regular dos separadores dos *splits* indicam que o caractere anterior ao '$' não pode ser um '\', o *split* faz a separação juntamente com esse caractere anterior diferente de '\'. Para deixar mais claro, o caractere anterior ao '$' era considerado como parte do separador e, então, o deixava de fora do vetor (o *split* não coloca os separadores como parte do vetor), fazendo com que faltasse o último caractere de todos os índices dos vetores gerados, como é ilustrado na Figura 13, onde os caracteres ":" e "2" deveriam ser o último caractere dos conteúdos dos índices 0 e 1, respectivamente, mas são perdidos após o *split*.

■ String antes do *split:*

```
<p>For example:$$\int_0^1(15-x)^2(\text{d$2\$$}x)^2$$</p>&#xA;
```

■ Vetor após o *split*:

0	For example
1	\int_0^1(15-x)^2(\text{d$2\$$}x)^
2

Figura 13 - Exemplo do problema da perda do último caractere dos índices após o *split*

Para resolver este problema, todos os caracteres '$' foram previamente substituídos por ' $' (um espaço em branco foi inserido precedendo o '$') e depois, para não separar os identificadores '$$', e nem os '$' dos '\' precedentes, todos os trechos '$ $' foram substituídos por '$$' e os trechos '\ $' substituídos por '\$'. Todas as substituições foram feitas utilizando o comando *Java replace*. O mesmo exemplo da Figura 13 foi utilizado para mostrar as correções feitas (Figura 14) e o vetor obtido na execução do *split* após a correção da *String* (Figura 15), onde podem ser observados os espaços em braço antes dos caracteres '$', os caracteres '$$' e o escape '\$' em suas formas corretas.

■ Utilizando o comando *replace*:

- Substituir os caracteres '$' por ' $';

```
<p>For example: $ $\int_0^1(15-x)^2(\text{d $2\ $ $}x)^2 $ $</p>&#xA;
```

- Substituir os trechos '\ $' por '\$';

```
<p>For example: $ $\int_0^1(15-x)^2(\text{d $2\$ $}x)^2 $ $</p>&#xA;
```

- Substituir os trechos ' $ $' por ' $$';

```
<p>For example: $$\int_0^1(15-x)^2(\text{d $2\$ $}x)^2 $$</p>&#xA;
```

Figura 14 - Correções realizadas para o problema da perda do última caractere

- String corrigida antes do *split:*

For example: $$\int_0^1(15-x)^2(\text{d $2\$ $}x)^2 $$

- Vetor após o *split*:

0	For example:
1	\int_0^1(15-x)^2(\text{d $2\$ $}x)^2
2

Figura 15 - Execução correta do *split* após correção da *String* inicial

4.2.3.1 O problema das fórmulas inseridas nos elementos textuais

Como já descrito no capítulo Referencial Teórico, fórmulas podem ser inseridas em elementos textuais que já estão dentro de uma fórmula. Essa possibilidade gera problemas no momento da separação do conteúdo em um vetor. O comando *split* apenas divide o conteúdo em índices de acordo com o separador definido para ele, independentemente de onde esteja o separador, pois, obviamente, não consegue identificar se está dentro de um comando de elemento textual ou não, já que apenas percorre a *String* procurando a ocorrência do separador. Dessa forma, observa-se na Figura 16 que os índices ímpares dos vetores gerados guardavam fórmulas incompletas, pois os mesmos eram finalizados com o identificador de trecho de fórmula de dentro do elemento textual, que na verdade está abrindo outro trecho de fórmula.

- String antes do *split*:

For example: $\int_0^1(15-x)^2(\text{a: } y + \text{d$ \sqrt{2} $}x)^2$

- Vetor após o *split*:

0	For example:
1	\int_0^1(15-x)^2(\text{a: } y + \text{d
2	\sqrt{2}
3	}x)^2
4

Figura 16 - Exemplo de vetor gerado sem considerar a existência dos elementos textuais

Para solucionar este problema, se um índice do vetor gerado possuir um caractere '{', indicando abertura de comando, o algoritmo verifica se existe um comando de elemento textual neste índice, possibilitando a existência do problema aqui descrito. Os comandos textuais previstos no algoritmo são '\text', '\mbox', '\fbox', '\raisebox' e '\hbox'. Se o índice contiver um desses comandos, é preciso verificar se existe um caractere '}' posterior, indicando fechamento deste comando. Se o fechamento não existir, significa que existe uma fórmula dentro deste comando. O algoritmo faz essa verificação contando o número de aberturas e fechamentos de comando que ocorrem após a abertura do comando de elemento textual, já que outros comandos também podem estar contidos no mesmo. Com o auxílio de um contador, iniciado com o valor um, a partir do índice em que ocorre o comando de elemento textual seguido de um caractere de abertura, a cada ocorrência do caractere '{' o contador é incrementado e, a cada ocorrência do caractere '}' o contador é decrementado. Se o contador chegar no valor zero, significa que o comando foi devidamente fechado. Se ao percorrer todo o restante do índice o contador continuar maior que zero, indica que existe uma fórmula dentro deste comando. Assim, o restante do comando estará nos próximos índices do vetor. O índice seguinte contém a fórmula inserida dentro do elemento textual pois, assim como o identificador de fórmula de abertura foi utilizado no *split*, o de fechamento também foi. Portanto, o fechamento do comando de elemento textual estará dois índices à frente do índice onde a abertura está. Neste momento o algoritmo insere a fórmula existente dentro do

comando no *ArrayList* de fórmulas, concatena os três índices em um só (inserindo novamente os caracteres '$' entre eles) e verifica a existência de novos comandos de elementos textuais no índice concatenado que possam gerar o mesmo problema. Ao final, insere o índice completo no *ArrayList*. Dessa forma, as fórmulas existentes dentro de comandos textuais serão extraídas de forma independente e as fórmulas que contém elementos textuais também serão extraídas, com todos os seus elementos completos (inclusive os caracteres '$'). O mesmo exemplo da Figura 16 está ilustrado na Figura 17, mas com o problema já tratado no algoritmo, onde, a fórmula contida no elemento textual e a fórmula contida no índice 1 do vetor serão extraídas de forma correta.

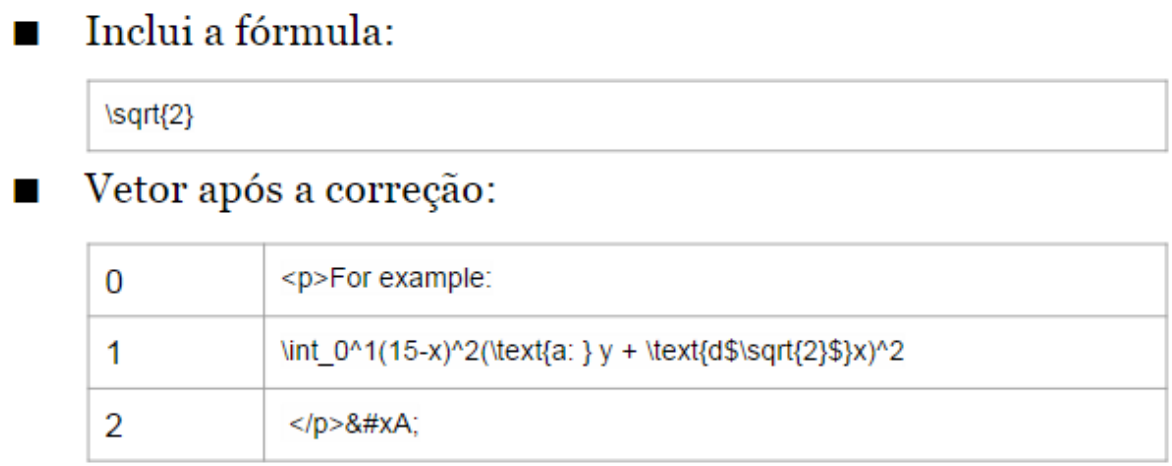

Figura 17 - Exemplo de vetor com o problema dos elementos textuais corrigido

4.2.3.2 O problema dos comentários em fórmulas

Como também foi descrito no capítulo Referencial Teórico, comentários podem ser inseridos em fórmulas e são identificados pelo caractere '%'. Os comentários em fórmulas devem ser desconsiderados e eliminados das mesmas, já que são trechos de texto irrelevantes para a busca. Como os comentários são iniciados com '%' e finalizados com uma quebra de linha, identificada pelo comando '\n', antes da inserção das fórmulas no *ArrayList*, o comando *Java ReplaceAll* substitui o caractere '%' e todos os seguintes até encontrar um '\n'. A definição do que deve ser substituído é feita através de expressão regular. A Figura 18 mostra um exemplo do problema e a sua solução.

- String com trecho de fórmula:

```
<p>For example: $\int_0^1(15-x)^2(\text{d$ \sqrt{2} %trecho de formula
$}x)^2$</p>&#xA;
```

- *String* após a correção:

```
<p>For example: $\int_0^1(15-x)^2(\text{d$ \sqrt{2} $}x)^2$</p>&#xA;
```

Figura 18 - Exemplo do problema dos comentários em fórmulas

Vale frisar que para os dois problemas relatados à cima, a possibilidade de existência de escape também foi considerada e que também foi definido na expressão regular que encontra os caracteres '{', '}' e '%' nos conteúdos que não os considerem caso sejam precedidos pelo escape '\'. Esta definição gera o mesmo problema da expressão regular que busca os caracteres '$', eliminando os caracteres precedentes. O problema foi resolvido também da mesma maneira, incluindo espaços em branco precedendo estes caracteres.

Após a execução dos processos descritos para todas as tuplas da tabela *tb_html* e inserção das fórmulas no *ArrayList*, as mesmas são inseridas na tabela *tb_equation*. Cada elemento existente no *ArrayList* gera um comando de inserção. Além disso, a relação entre a fórmula e a sua postagem correspondente é armazenada na tabela *rl_html_equation*, inserindo o *id* gerado sequencialmente na inserção da fórmula na tabela e o *id* da postagem correspondente, armazenado no *ArrayList* juntamente com a fórmula. Porém, se uma fórmula já existe na tabela *tb_equation*, ela não deve ser inserida novamente. O algoritmo recupera o *id* desta fórmula e apenas insere uma relação na tabela *rl_html_equation*, indicando que aquela fórmula ocorre naquela postagem.

Ao final da execução do algoritmo, os dados da extração são impressos na tela, como número de postagens e de equações extraídas.

O teste nesta etapa é feito verificando se os dados inseridos na tabela de fórmulas e suas postagens correspondentes conferem com a postagem na página *web* do fórum, constatando a existência das fórmulas no seu conteúdo. Além disso, é feita uma visualização de uma pequena porção das fórmulas diretamente na base de dados, procurando a existência de textos em seu interior e, em caso positivo, verificando a procedência dos mesmos (erro do usuário ou erro do algoritmo).

Após estes testes iniciais, o algoritmo tratador e analisador léxico, já desenvolvido anteriormente, é executado, tratando as fórmulas para casos de elementos *HTML* em seu interior, entre outros casos, e gerando a expressão léxica correspondente a cada fórmula. Esse execução gera uma lista de fórmulas que não estão completas, como o caso de um comando que não está fechado. Por isso, serve como teste final para esta etapa.

4.2.4 Etapa 4: Geração de gráfico e dados para análise dos resultados

Nesta etapa, um algoritmo para geração de dados e gráfico foi desenvolvido. Os dados gerados levam em conta o número de *tokens* (elementos) da expressão léxica de cada fórmula existente na tabela *tb_equation*. O algoritmo recupera da base de dados a expressão léxica e faz uma contagem dos *tokens* contidos na mesma. A partir desta contagem, gera um arquivo texto com todas as fórmulas que têm mais de 500 *tokens* em sua expressão léxica, bem como a postagem a que pertence. Além disso, o algoritmo também gera um arquivo texto com a lista do número de fórmulas que têm cada número de *tokens*. Por exemplo, com zero *tokens*, são X fórmulas, com um *token*, Y fórmulas, com dois *tokens* Z fórmulas, e assim por diante. Esse arquivo texto é exportado em modo *csv* para uma planilha onde, é calculada a distribuição de probabilidade em que cada número de *tokens* está associado. Por exemplo, com zero *tokens*, são 5% das fórmulas, com um *token*, 7%, e assim por diante. Um gráfico dessa lista é gerado, com a relação entre a distribuição de probabilidade e o número de fórmulas. Além disso, a porcentagem das fórmulas que têm mais de 70 *tokens* em sua expressão léxica é calculada.

O teste desta etapa é feito conferindo se as fórmulas que possuem mais de 500 *tokens* foram extraídas de forma correta, ou se tiveram algum erro (de usuário ou de algoritmo) e verificando se a soma do número de fórmulas na lista gerada é a mesma do número de fórmulas que foram extraídas.

5
Resultados

Este capítulo apresenta os resultados obtidos após o desenvolvimento e execução dos algoritmos aplicando-os nas bases de dados trabalhadas.

5.1 Números das importações das bases de dados

5.1.1 *Mathematica*

Na importação do fórum *Mathematica*, foram obtidas 57393 postagens, das quais apenas 8220 continham equações. Foram encontrados 36312 trechos de fórmulas nas suas postagens, que geraram 22134 fórmulas distintas inseridas na tabela *tb_equation*, das quais 2613 têm mais de 70 *tokens* em sua expressão léxica, representando 11,81%. Na Figura 11, pode-se observar o gráfico gerado na Etapa 4.

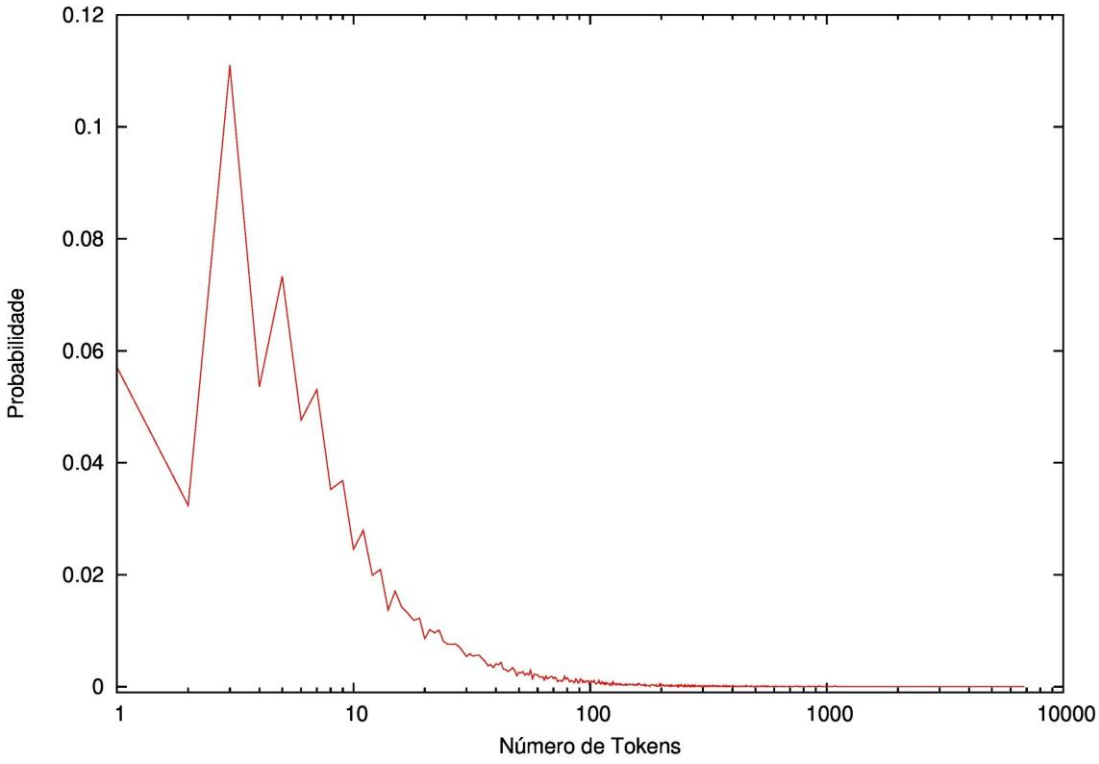

Figura 19 - Gráfico gerado na etapa 4 para o fórum *Mathematica*

5.1.2 *Mathematics*

Na importação do fórum *Mathematics*, foram obtidas 1002727 postagens, das quais 865241 continham equações. Foram encontrados 8878944 trechos de fórmulas nas suas postagens, que geraram 3679321 fórmulas distintas inseridas na tabela *tb_equation*, das quais 69098 têm mais de 70 *tokens* em sua expressão léxica, representando 1,88%. Na Figura 12, pode-se observar o gráfico gerado na Etapa 4.

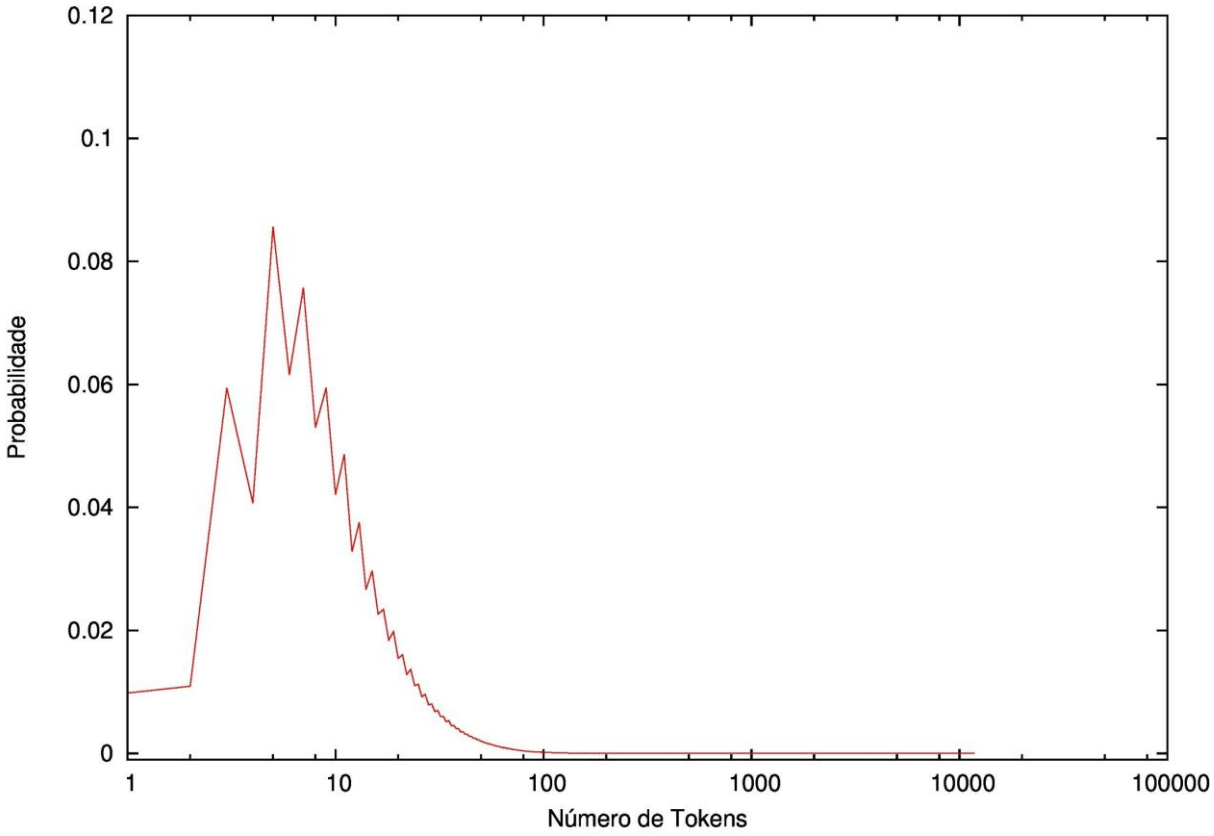

Figura 20 - Gráfico gerado na etapa 4 para o fórum *Mathematics*

5.1.3 *MathOverFlow*

Na importação do fórum *MathOverFlow*, foram obtidas 156407 postagens, das quais 109485 continham equações. Foram encontrados 1583382 trechos de fórmulas nas suas postagens, que geraram 578018 fórmulas distintas inseridas na tabela *tb_equation*, das quais 2989 têm mais de 70 *tokens* em sua expressão léxica, representando 0,5%. Na Figura 13, pode-se observar o gráfico gerado na Etapa 4.

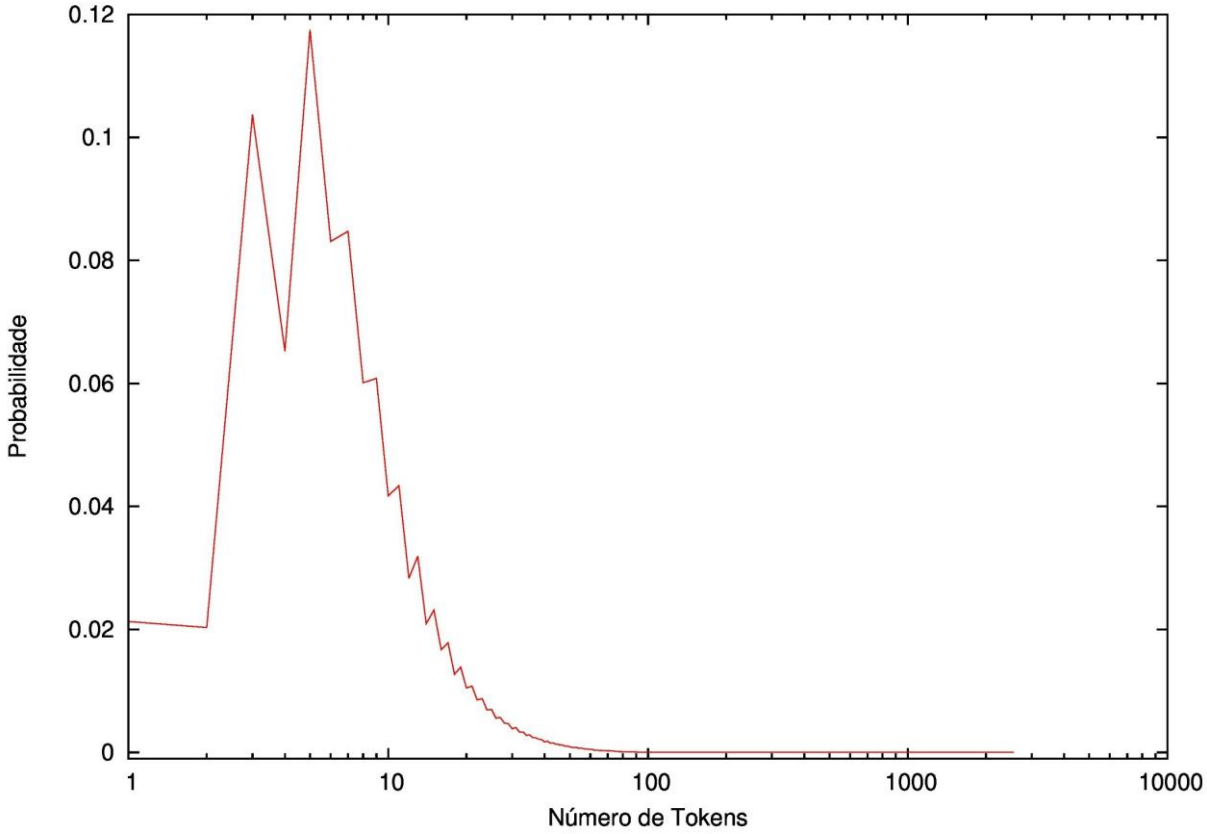

Figura 21 - Gráfico gerado na etapa 4 para o fórum *MathOverFlow*

5.2 Fórum *Mathematica* desconsiderado

Após executar os algoritmos desenvolvidos para a base de dados do fórum *Mathematica* e obter os resultados gerados, foi observado que a quantidade de fórmulas que possuem mais de 70 *tokens* em sua expressão léxica é de mais de 11% da quantidade de fórmulas extraídas das postagens do fórum, enquanto que nos outros dois fóruns esse número não passou de 2%. Um número elevado de *tokens* geralmente indica a existência de trechos textuais no interior da fórmula. Com isso, foram analisadas as fórmulas que possuem mais de 500 *tokens* e na maioria dos casos a existência de trechos textuais foi confirmada.

Observando os gráficos ilustrados na seção 5.1, nota-se que para 70 *tokens* a curva está pouco íngreme, diferentemente dos gráficos dos outros dois fóruns, indicando que o número de fórmulas com 70 ou mais *tokens* é elevado, comparando ao número total de fórmulas.

Por se tratar de um fórum voltado para usuários de um programa matemático, muitos usuários inseriam comandos específicos do programa que continham o caractere '$' em suas postagens, mas não o precediam do caractere de escape, causando aberturas e fechamentos de trechos de fórmulas indesejados, gerando muitos erros na extração de fórmulas.

A partir destas constatações, considerando a possibilidade de existência de trechos textuais em fórmulas de menor número de *tokens*, analisando a baixa quantidade de fórmulas extraídas (comparando com os outros dois fóruns) e notando que a maioria das postagens se referem a assuntos relacionados ao programa matemático, e não a problemas matemáticos em si, concluiu-se que a importação do fórum *Mathematica* não se faz necessária.

Portanto, os resultados obtidos para o fórum *Mathematica* foram desconsiderados e não serão incluídos na ferramenta *SearchOnMath*.

6
Conclusões e Trabalhos Futuros

Este capítulo expõe as conclusões que puderam ser tiradas neste trabalho a partir dos objetivos e resultados, além de indicar possíveis trabalhos futuros a serem desenvolvidos a partir deste.

Observando os resultados obtidos, conclui-se que o desenvolvimento dos algoritmos de importação das bases de dados e extração de fórmulas de fóruns *Stack Exchange* foi satisfatório, pois a quantidade de novas fórmulas obtidas foi alta, sendo uma porcentagem baixa de fórmulas com quantidade elevada de *tokens*, indicando menor possibilidade de ocorrências de trechos textuais dentro de fórmulas.

Com o desenvolvimento deste trabalho, a atualização das bases de dados dos fóruns deste padrão se tornaram fáceis e práticas, com o tempo de execução extremamente ligado à quantidade de postagens existente em cada uma. Sendo assim, a ferramenta *SearchOnMath* ganha em abrangência e qualidade em suas pesquisas, já que passará a oferecer suporte a mais bibliotecas.

Além disso, os algoritmos desenvolvidos são generalizados, ou seja, não possuem especificidades para cada base de dados, sendo funcional para todas as bases que trabalham no padrão *Stack Exchange*.

Contanto, ainda existem melhorias que podem ser trabalhadas futuramente nestes algoritmos, como criação de heurísticas para melhorar a identificação de trechos textuais no interior das fórmulas extraídas, unificação dos algoritmos, possibilitando que todo o processo seja realizado com apenas uma execução, desenvolvimento de interface gráfica interativa onde a importação pode ser realizada apenas arrastando os arquivos *XML* para a tela ou apontando o local onde estão.

7
Referências Bibliográficas

Este capítulo lista as referências bibliográficas que apoiaram o desenvolvimento deste trabalho.

Asperti, A., Guidi, F., Coen, C. S., et˜al. *A Content Based Mathematical Search Engine: Whelp, Lecture Notes in Computer Science Types for Proofs and Programs*, v. 3839, pp. 17–32, 2006.

Avny, M., Arnon, A., Alyshayev, L. *Symbolab Scientific Equation Search*. http://symbolab.com/, 2013.

Brin, S., Page, L. *The anatomy of a largescale hypertextual Web search engine.* WWW7: Proceedings of the seventh international conference on World Wide Web 7, pp. 107–117, Amsterdam, The Netherlands, 1998. Elsevier Science Publishers B. V. doi: http://doi.acm.org/10.1145/2484028.2484083.

Gonzaga, F. B. *Recuperação de Informação Orientada ao Domínio da Matemática*, Tese de Doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013

Kamali, S., Tompa, F. W. 2013. *Retrieving documents with mathematical content. Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '13). ACM, New York, NY, USA, 353362. DOI=10.1145/2484028.2484083 http://doi.acm.org/10.1145/2484028.2484083

Kohlhase, M., Sucan, I. *A Search Engine for Mathematical Formulae, Lecture Notes in Computer Science Artificial Intelligence and Symbolic Computation*, v. 4120, pp. 241–253, 2006.

Oreskovic, A., Chang, R. *Google widens lead in U.S. searches: comScore.* http://www.reuters.com/article/idUSTRE53E6YT20090415, 2009.

Shatnawi, M., Youssef, A. *Equivalence detection using parsetree normalization for math search, 2nd International Conference on Digital Information Management*, v. 2, pp. 643–648, 2007.

Stalnalker, D., Zanibbi, R. *Math expression retrieval using an inverted index over symbol* pairs (*SPIE* 9402 '15). Document Recognition and Retrieval XXII, 940207 DOI:10.1117/12.2074084; http://doi.acm.org/10.1145/2484028.2484083

Xiaoyan, L., Liangcai, G., Xuan, H., Zhi, T., Yingnan, X., Xiaozhong, L. 2014. *A mathematics retrieval system for formulae in layout presentations. Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (SIGIR '14). ACM, New York, NY, USA, 697706. DOI=10.1145/2600428.2609611 http://doi.acm.org/10.1145/2600428.2609611

Xuan, H., Liangcai, G., Xiaoyan, L., Zhi, T., Xiaofan, L., Baker, J. B. 2013. *WikiMirs: a mathematical information retrieval system for wikipedia. Proceedings of the 13th ACM/IEEECS joint conference on Digital libraries* (JCDL '13). ACM, New 10 York, NY, USA, 1120. DOI=10.1145/2467696.2467699 http://doi.acm.org/10.1145/2467696.2467699

8
Anexos

8.1 Anexo I - Estrutura de tabelas geradas e alimentadas à partir de *Dump* do *Stack Exchange*

Descrição das tabelas e de seus campos presentes no *Dump*, bem como a relação entre as mesmas.

8.1.1 Tabela *badges*:

Contém o "crachá" ou papel ou tipo de cada usuário cadastrado (Tabela *users*);

- ***Id*:** Identificador da tupla, PK;
- ***UserId*:** Referencia o *Id* do usuário correspondente, ou seja, é uma chave estrangeira para o *Id* de uma tupla da tabela *users*;
- ***Name*:** Nome do "crachá" do usuário. Ex: *Student*, *Teacher*;
- ***CreationDate*:** Data e hora da criação da tupla.

8.1.2 Tabela *comments*:

Contém os comentários inseridos por usuários em postagens (Tabela *posts*);

- ***Id*:** Identificador da tupla, PK;
- ***PostId*:** Referencia a postagem que recebeu o comentário, através do seu *Id*. Ou seja, é uma chave estrangeira para o Id de uma tupla da tabela *posts*;
- ***Score*:** Número de "pontos" ou "votos" que o comentários recebeu;
- ***Text*:** Conteúdo completo do comentário;
- ***CreationDate*:** Data e hora da criação da tupla;

- ***UserDisplayName*:** Nome de exibição do usuário que inseriu o comentário. Somente preenchido se o *UserId* não existe mais na tabela *users*.
- ***UserId*:** Referencia o *Id* do usuário que inseriu o comentário, ou seja, é uma chave estrangeira para o Id de uma tupla da tabela *users*. Se o usuário referente a este *Id* foi excluído, não é preenchido.

8.1.3 Tabela *post_history*:

Contém o histórico de alterações das postagens (incluindo a criação), podendo haver mais de um histórico por postagem. É dividido em título, corpo e *tags* de uma postagem, identificados pelo campo *PostHistoryTypeId*.

- ***Id*:** Identificador da tupla, PK;
- ***PostHistoryTypeId*:** Identificador do tipo de histórico de alteração. A descrição de cada um está no Anexo II;
- ***PostId*:** Referencia a postagem, através do seu *Id*. Ou seja, é uma chave estrangeira para o Id de uma tupla da tabela *posts*;
- ***RevisionGUID*:** Chave de versão da postagem após as alterações. Duas ou mais alterações seguidas feitas pelo mesmo usuário são inseridas na mesma versão.
- ***CreationDate*:** Data e hora da criação da tupla;
- ***UserId*:** Referencia o *Id* do usuário que realizou a alteração (ou criação), ou seja, é uma chave estrangeira para o *Id* de uma tupla da tabela *users*;
- ***UserDisplayName*:** Nome de exibição do usuário que realizou a alteração. Preenchido somente se o usuário foi removido e não existe mais referência para o *UserId* na tabela *users*;
- ***Comment*:** Este campo contém um comentário inserido pelo usuário que realizou a edição da postagem. Se *PostHistoryTypeId* = 10, contém o *CloseReasonId* do motivo do fechamento da postagem. A lista dos *CloseReasonId's* é:

  CloseReasonId's antigas:

  1 - *Exact Duplicate* (Duplicação Exata);

2 - *Off-topic* (Questão que não se enquadra na ideia do fórum. Ex.: Questão sobre Geografia em um fórum de Matemática);

3 - *Subjective and argumentative* (Subjetivo e argumentativo);

4 - *Not a real question* (Não é uma questão real);

7 - *Too localized* (Muito localizada. A questão envolvia uma quantidade muito pequena de possíveis usuários, sendo muito localizada geograficamente, onde a questão se refere a uma rua de uma cidade qualquer, ou também muito localizada na questão temporal, onde em um pequeno espaço de tempo a questão já não será útil);

10 - *General Reference* (Questão extremamente trivial, que pode ser solucionada com uma simples consulta ao manual, por exemplo);

20 - *Noise or pointless* (Somente Sites Meta. Questão que não acrescenta nada de útil ao fórum. Fechada por ser considerada uma "distração" às demais questões mais relevantes).

CloseReasonId's atuais:

101 - *Duplicate* (Questão Duplicada);

102 - *Off-topic* (Questão que não se enquadra na idéia do fórum. Ex.: Questão sobre Geografia em um fórum de Matemática);

103 - *Unclear what you're asking* (A questão não está clara);

104 - *Too broad* (Questão muito ampla);

105 - *Primarily opinion-based* (Questão que gera opiniões pessoais);

- ***Text*:** Uma versão crua do novo valor para a determinada versão. Conteúdo editado (ou original) do Título, Corpo ou *Tags* da postagem.

  Se *PostHistoryTypeId* = 10, 11, 12, 13, 14, ou 15 essa coluna conterá uma *string JSON* codificada com todos os usuários que votaram para a alteração do tipo *PostHistoryTypeId*;

  Se *PostHistoryTypeId* = 17 essa coluna conterá detalhes da migração, como um "from <url>" (*url* origem) e um "to <url>"(*url* destino);

8.1.4 Tabela *post_links*:

Contém a relação de *links* para outras postagens em cada postagem, podendo haver mais de um *link* por postagem. Existem dois tipos de *links*, *Linked* e *Duplicate*, identificados na coluna *PostLinkTypeId*.

- ***Id*:** Identificador da tupla, PK;
- ***CreationDate*:** Data e hora de criação da tupla;
- ***PostId*:** Identificador da postagem de origem;
- ***RelatedPostId*:** Identificador da postagem à qual o *link* referencia;
- ***LinkTypeId*:** Identificador do tipo de *link*, o qual:

1 - *Linked* (*Link* presente no corpo de uma postagem que referencia outra postagem);

3 - *Duplicate* (*Link* entre duas postagens duplicadas. Redireciona para a postagem destino quando a *URL* da postagem origem é acionada);

8.1.5 Tabela *posts*:

Contém a relação de todas as postagens, podendo ser questão ou resposta;

- ***Id:*** Identificador da tupla, PK;
- ***PostTypeId*:** Identificador do tipo de postagem, podendo ser:

1 - *Question* (Questão/Pergunta);

2 - *Answer* (Resposta);

3 - *Wiki*;

4 - *Tag wiki excerpt* (Descrição Resumida de uma *tag*);

5 - *Tag wiki* (Descrição completa de uma *tag*);

6 - *Moderator nomination* (Candidatura à moderação - usuários fazem postagens se apresentando e explicando o porquê deve ser eleito moderador);

7 - *Wiki placeholder* (Postagem informativas aos usuários, como FAQ, descrição de eleição de moderadores, etc);

8 - *Privilege* Wiki (Postagem informativa sobre privilégios);

- ***AcceptedAnswerId*:** Somente é preenchido se o *PostTypeId* for *"1"*, ou seja, se a postagem for uma pergunta. Identifica a postagem do tipo resposta que foi escolhida como a melhor resposta.
- ***ParentId*:** Somente é preenchido se o *PostTypeId* for *"2"*, ou seja, se a postagem for uma resposta. Identifica a postagem pai que contém essa resposta, ou seja, é uma chave estrangeira para o *Id* de uma tupla da mesma tabela *posts*;
- ***Score*:** Pontuação da postagem;
- ***ViewCount*:** Contador de visualizações da postagem;
- ***Body*:** Conteúdo do corpo da postagem;
- ***OwnerUserId*:** Identificador do usuário que é proprietário da postagem. É uma chave estrangeira para um *Id* de uma tupla da tabela *users*;
- ***OwnerDisplayName*:** Nome de exibição do usuário proprietário da postagem. Somente preenchido quando o *OwnerUserId* não existe mais na tabela *users*.
- ***LastEditorUserId*:** Identificador do usuário que realizou a última alteração na postagem. É uma chave estrangeira para um *Id* de uma tupla da tabela *users*;
- ***LastEditorDisplayName*:** Nome de exibição do usuário que realizou a última edição na postagem. Somente preenchido quando o *LastEditorUserId* não existe mais na tabela *users*.
- ***LastEditDate*:** Data e hora da última edição realizada na postagem;
- ***LastActivityDate*:** Data e hora da última atividade realizada na postagem;
- ***Title*:** Título da postagem;
- ***Tags*:** *Tags* presentes na postagem;
- ***AnswerCount*:** Contador de respostas postadas para esta postagem;
- ***CommentCount*:** Contador de comentários postados para esta postagem;
- ***FavoriteCount*:** Contador de usuários que favoritaram a postagem (somente em postagens do tipo questão, ou seja, em que *PostTypeId = 1*);
- ***CreationDate*:** Data e hora de criação da tupla;

- ***ClosedDate*:** Data e hora de fechamento da postagem. Somente presente se a postagem foi fechada.
- ***CommunityOwnedData*:** Data e hora em que a postagem foi transformada em propriedade da comunidade. Somente presente se a postagem foi transformada em propriedade da comunidade.

8.1.6 Tabela *tags*:

Contém as *tags* existentes no sistema. *Tags* podem ser vistas como categorias e/ou filtros que podem ser marcadas nas postagens.

- ***Id*:** Identificador da tupla, PK;
- ***TagName*:** Nome da *tag*;
- ***Count*:** Quantidade de postagens marcadas com a *tag*;
- ***ExcerptPostId*:** Identificador da postagem do tipo *Tag Wiki Excerpt*, ou seja, é uma postagem de descrição resumida da *tag*. Chave estrangeira para um Id de uma tupla da tabela *posts*.
- ***WikiPostId*:** Identificador da postagem do tipo *Tag Wiki*, ou seja, é uma postagem de descrição completa da *tag*. Chave estrangeira para um *Id* de uma tupla da tabela *posts*.

8.1.7 Tabela *users*:

Contém a relação de usuários cadastrados no sistema.

- ***Id*:** Identificador da tupla, PK;
- ***Reputation*:** Reputação que o usuário tem no sistema;
- ***CreationDate*:** Data e hora da criação da tupla;
- ***DisplayName*:** Nome de exibição do usuário;
- ***LastAcessDate*:** Data e hora do último acesso do usuário ao sistema;

- ***Views*:** Quantidade de visualizações nas postagens do usuário;
- ***WebSiteUrl*:** *URL* do *WebSite* do usuário cadastrado pelo mesmo;
- ***Location*:** Localidade do usuário (Cidade, Estado, País, etc);
- ***AboutMe*:** Autodescrição escrita pelo usuário;
- ***Age*:** Idade do usuário;
- ***UpVotes*:** Quantidade de votos positivos recebidos nas postagens do usuário;
- ***DownVotes*:** Quantidade de votos negativos recebidos nas postagens do usuário;
- ***AccountId*:** Identificador do perfil do usuário na rede *Stack Exchange*;
- ***EmailHash*:** Sempre nulo;

8.1.8 Tabela *votes*:

Contém os votos e seus tipos recebidos em cada postagem.

- ***Id*:** Identificador da tupla, PK;
- ***PostId*:** Identificador da postagem que recebeu o voto, ou seja, é uma chave estrangeira para um *Id* de uma tupla da tabela *posts*.
- ***VoteTypeId*:** Identificador do tipo de voto, podendo ser:

  1 - *AcceptedByOriginator* (Resposta aceita pelo proprietário da questão);

  2 - *UpMod* (Postagem positivada);

  3 - *DownMod* (Postagem negativada);

  4 - *Offensive* (Postagem ofensiva);

  5 - *Favorite* (Postagem Favoritada) - Se *VoteTypeId* = 5 *UserId* deve ser preenchido;

  6 - *Close* (Postagem deve ser fechada);

  7 - *Reopen* (Postagem deve ser reaberta);

  8 - *BountyStart* (Postagem deve ter sua contagem de recompensa iniciada) - Se *VoteTypeId* = 8, *UserId* e *BountyAmount* devem ser preenchidos;

  9 - *BountyClose* (Postagem deve ter sua contagem de recompensa encerrada) - Se *VoteTypeId* = 9, *BountyAmount* deve ser preenchido;

10 - *Deletion* (Postagem deve ser excluída);
11 - *Undeletion* (Postagem deve ser restaurada);
12 - *Spam* (Postagem é Spam);
15 - *ModeratorReview* (Postagem deve ser revisada por um moderador);
16 - *ApproveEditSuggestion* (Sugestão de Edição deve ser aprovada);

- ***UserId*:** Identificador do usuário que "favoritou" a postagem. Chave estrangeira para um Id de uma tupla da tabela *users*. Preenchido somente se *VoteTypeId* for *"5"* ou *"8"*;
- ***CreationDate*:** Data e hora da criação da tupla;
- ***BountyAmount*:** Total da recompensa recebida pelo usuário por sua postagem. Preenchido somente se *VoteTypeId* for *"9"*;

8.2 Anexo II - Descrição de cada *Id* possível no campo *PostHistoryTypeId* da tabela *post_history*

Tabela 7 - Descrição de cada *Id* possível no campo *PostHistoryTypeId* da tabela *post_history*

Id	**Título**	**Descrição**
1	*Initial Title*	O título da postagem original.
2	*Initial Body*	O corpo da postagem original.
3	*Initial Tags*	As *tags* presentes na postagem original.
4	*Edit Title*	Alteração no título da postagem.
5	*Edit Body*	Alteração no corpo da postagem. O texto bruto é armazenado aqui como *markdown*.
6	*Edit Tags*	Alteração nas *tags* da postagem.
7	*Rollback Title*	O título da postagem foi revertido para uma versão anterior.
8	*Rollback Body*	O corpo da postagem foi revertido para uma versão anterior - o texto bruto é armazenado aqui.
9	*Rollback Tags*	As *tags* da postagem foram revertidas para uma versão anterior.
10	*Post Closed*	A postagem foi votada para ser fechada.
11	*Post Reopened*	A postagem foi votada para ser reaberta.
12	*Post Deleted*	A postagem foi votada para ser removida.
13	*Post Undeleted*	A postagem foi votada para ser restaurada.

14	*Post Locked*	A postagem foi trancada pelo moderador.
15	*Post Unlocked*	A postagem foi destrancada pelo moderador.
16	*Community Owned*	A postagem foi transformada em propriedade da comunidade.
17	*Post Migrated*	A postagem foi migrada.
18	*Question Merged*	A questão teve outra questão mesclada em seu conteúdo.
19	*Question Protected*	A questão foi protegida por um moderador.
20	*Question Unprotected*	A questão foi desprotegida por um moderador.
21	*Post Disassociated*	Um administrador removeu o *OwnerUserId* da postagem.
22	*Question Unmerged*	Uma questão anteriormente mesclada teve suas respostas e votos restaurados.
24	*Suggested Edit Applied*	Sugestão de Edição na postagem foi aplicada.
25	*Post Tweeted*	A postagem foi "*tweetada*".
31	*Comment discussion moved to chat*	A discussão existente nos comentários foi movida pra o *chat*.
33	*Post Notice Added*	Notificação adicionada à postagem.
34	*Post notice removed*	Notificação removida da postagem.
35	*Post migrated away (replaces id 17)*	Postagem foi migrada para outro fórum (substituiu o *id* 17).
36	*Post migrated here (replaces id 17)*	Postagem foi migrada de outro fórum para o atual (substituiu o *id* 17).

37	*Post merge source*	Postagem foi mesclada com outra postagem (origem - postagem é desativada).
38	*Post merge destination*	Uma postagem foi mesclada com a postagem atual (destino - postagem se mantém ativa).