UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFAEL DE OLIVEIRA SERAFINI

Comparação entre métodos de extração de dados baseados na redundância de conteúdo

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Renata Galante
Co-orientador: MsC. Edimar Manica

Porto Alegre
2015

"*Saruman acredita que somente um grande poder*
pode segurar o mal, mas isso não é o que descobri.
Eu descobri que são os pequenos feitos de pessoas comuns
que mantém a escuridão afastada, pequenos atos de bondade e amor."
– Gandalf, o Cinzento

AGRADECIMENTOS

Agradeço, primeiramente, a meus pais, meus primeiros professores.

Minha mãe, Satira, que não reside mais no plano material, espero que receba meu agradecimento e veja que o filho que entregou ao mundo segue buscando ser alguém melhor.

Meu pai, Paulo, espero que se orgulhe, pois, se chego ao fim desta etapa, foi porque segui seu exemplo.

Agradeço ao meu irmão, André, que me ajuda a levar adiante o carinho da minha mãe, os ensinamentos do meu pai e o amor de e por ambos.

À minha família, Vargas e Serafini, por me receberem sempre com um carinho maior do que posso compreender, ao qual procuro retribuir, se não com o mesmo, com mais.

Agradeço a todos os meus amigos, que compartilharam minhas alegrias e meus problemas, caminharam comigo e que me fazem querer continuar sempre.

À minha namorada, Cecília, e a sua família, que me acolheram e me ajudaram nos últimos passos dessa jornada.

Aos colegas da Academia Wu Song, especialmente ao professor Rafael, que mantém esse espaço, onde posso esquecer os desafios da universidade e treinar meu espírito.

Agradeço à minha orientadora, Renata, e ao meu co-orientador, Edimar, que me ajudaram a dar o último passo com o nível de qualidade que a UFRGS mantém.

Por fim, agradeço aos professores que tive, principalmente àqueles que demonstraram sua paixão pelo ofício de educar.

Até então, terminar o ensino superior foi meu maior desafio e minha maior vitória. Essa vitória pertence a todos vocês que estiveram ao meu lado.

RESUMO

Instâncias de entidades do mundo real podem ser representadas em páginas da Web, chamadas de páginas-instância. A extração de dados em páginas-instância da Web visa extrair conhecimento útil para diversas aplicações, tais como, Google Calendar, Reverb e Scrapy. Sites com páginas-instância do mesmo domínio possuem redundância de conteúdo, ou seja, publicam instâncias ou atributos em comum. O objetivo deste trabalho é comparar três métodos de extração de dados baseados na redundância de conteúdo da Web. Duas bases de dados reais são usadas para testar os métodos, sendo uma delas criada neste trabalho. Os resultados de um dos métodos foram obtidos do artigo que o descreve. Os resultados dos outros métodos foram obtidos neste trabalho. Para isso foi obtida a implementação e adaptada para as bases de dados. Os resultados são comparados em termos de qualidade e eficiência. É demonstrado, através da comparação, que a estrutura das bases de dados afeta a qualidade da extração dos métodos de acordo com suas características. A notação usada para representar a posição de um atributo em uma página e o uso de redundância em nível de instância são exemplos dessas características. Os resultados podem ser usados para auxiliar a escolha de um método de extração, de acordo com a base de dados, e guiar a criação de novos métodos de extração.

Palavras-Chave: Extração de dados, redundância de conteúdo, páginas-instância.

Comparison between data extraction methods based on content redundancy

ABSTRACT

Instances of real world entities can be represented by Web pages, called instance-pages. The data extraction from instance-pages aims to extract knowledge through information useful for a number of applications, such as Google Calendar, Reverb and Scrapy. Websites with instance-pages from the same domain have content redundancy, that is, they publish instances or attributes in common. This work goal is to compare three data extraction methods based on Web content redundancy. Two real databases are used to evaluate the methods, being one of them created in this work. The results of one method were obtained from the paper that describes it. The results of the other methods were generated in this work. For that, the implementation was obtained and adapted to the databases. The results are compared in terms of quality and efficiency. It is demonstrated, through the comparison, that the structure of the databases affects the quality of the methods extraction according to its characteristics. The results can be used to help choosing an extraction method, according to the database, and guide the criation of new extraction methods.

Keywords: Data extraction, content redundancy, instance-pages.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AERD	Attribute Extraction based on Redundancy Detection
DOM	Document Object Model
FindAttrPos	Find Attribute Position
HTML	HyperText Markup Language
WEIR	Web-Extraction and Integration of Redundant data
XML	eXtensible Markup Language
XPath	XML Path Language

SUMÁRIO

1 INTRODUÇÃO

A Web é uma fonte de informações e o uso de seus dados é uma oportunidade de criar conhecimento. Para um melhor uso e manipulação, essas informações devem ser extraídas da Web e inseridas em um banco de dados onde podem ser comparadas. Segundo Gibson et al. (2005), uma fração significante de páginas da Web, 40-50%, são dinamicamente geradas populando páginas baseadas em *template*, ou seja, páginas com uma estrutura fixa. As páginas que descrevem uma instância de uma entidade do mundo real são chamadas páginas-instância. Por exemplo, uma página que descreve um projeto de software no site OW2[1]. Diferentes sites de um mesmo domínio possuem redundância de conteúdo. A redundância de conteúdo, no contexto deste trabalho, pode ser de dois tipos principais. A redundância em nível de instância ocorre quando dois ou mais sites publicam um conjunto de instâncias em comum e a redundância em nível de atributo ocorre quando dois ou mais sites publicam um conjunto de atributos em comum.

Aplicações que necessitam da extração de dados da Web incluem o *Scrapy*[2], que aplica a extração em teste automatizado e mineração, e o *ReVerb*[3], que extrai relações de sentenças em inglês. Recentemente o Google Calendar[4] iniciou um serviço de criação automática de eventos a partir de emails estruturados. O serviço identifica a estrutura de um recibo de passagem aérea enviado por e-mail para o usuário, por exemplo, e extrai as informações de destino e horário. O evento é criado com base nas informações extraídas do e-mail.

A extração de dados dos sites pode permitir a criação de uma extensa base de dados, mas para realizar extrações em larga escala são necessários métodos que não necessitem de constante intervenção humana. Assim, métodos que passam por um processo de treinamento automático para

[1] http://www.ow2.org/
[2] http://scrapy.org/
[3] http://reverb.cs.washington.edu/
[4] https://support.google.com/calendar/answer/6084018?hl=pt

identificar a estrutura de um site, e aprender as posições onde se encontram os dados relevantes, são uma alternativa.

O objetivo deste trabalho é comparar métodos de extração de dados baseados em redundância de conteúdo da Web, analisando as vantagens e desvantagens de cada um de acordo com o tipo de estrutura adotado em diferentes bases de dados. Uma base foi criada, com seu gabarito, neste trabalho.

Os testes com os métodos demonstraram que determinadas características dos métodos, como a notação de caminho com índice, produzem resultados melhores em sites em que não há variação da posição dos atributos. Nos sites onde há variação de posição dos atributos, a notação de caminho sem índice, aliada com um filtro por rótulo (nome que o atributo recebe em cada site), produz uma extração com maior qualidade. Além disso, os métodos que utilizam a redundância em nível de instância são mais robustos na busca por valores referentes à instância descrita pela página de extração.

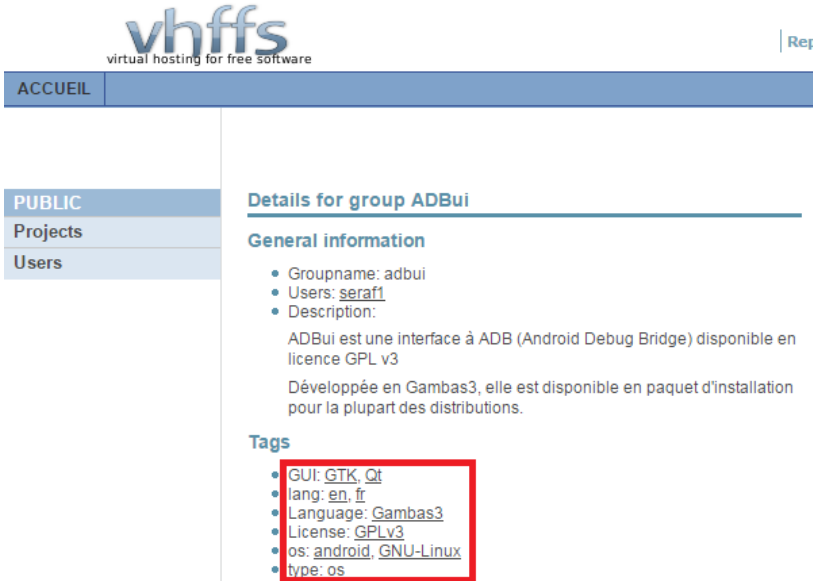
O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os conceitos básicos necessários para o entendimento deste trabalho. O Capítulo 3 apresenta os métodos de extração de dados baseados em redundância de conteúdo da Web estudados neste trabalho. O Capítulo 4 descreve a avaliação experimental e compara os resultados dos testes dos métodos. Por fim, o Capítulo 5 apresenta a conclusão do trabalho.

2 CONCEITOS BÁSICOS

Este capítulo apresenta os conceitos básicos e os trabalhos sobre extração de dados da Web baseado em redundância de conteúdo.

Reaproveitando os conceitos clássicos de banco de dados, conforme Heuser (2009), define-se **entidade** como um "conjunto de objetos da realidade modelada sobre os quais deseja-se manter informações [...]", **atributo** é um "dado que é associado a cada ocorrência de uma entidade [...]" e **instância** é uma ocorrência de um objeto particular. Estes conceitos foram mapeados para páginas Web que serão usadas para extração de dados. Por exemplo, a Figura 2.1 mostra uma instância da entidade projeto de software, com os atributos Linguagem de Programação (*Language*), Licença (*License*) e Sistema Operacional (*os*), entre outros.

Figura 2.1: Instância de um projeto de software



Fonte: *http://projects.tuxfamily.org/?do=group;name=adbui*

Os atributos podem receber uma denominação diferente em cada site, essa denominação é chamada de **rótulo** (*alias*). Na página da Figura 2.2, do site PGfoundry[5], o atributo situação de desenvolvimento recebe o rótulo "*Development Status*" mas pode receber outro rótulo em outro site.

Figura 2.2: Rótulo referente ao atributo Situação de Desenvolvimento e valor respectivo

Fonte: Elaborado pelo autor.

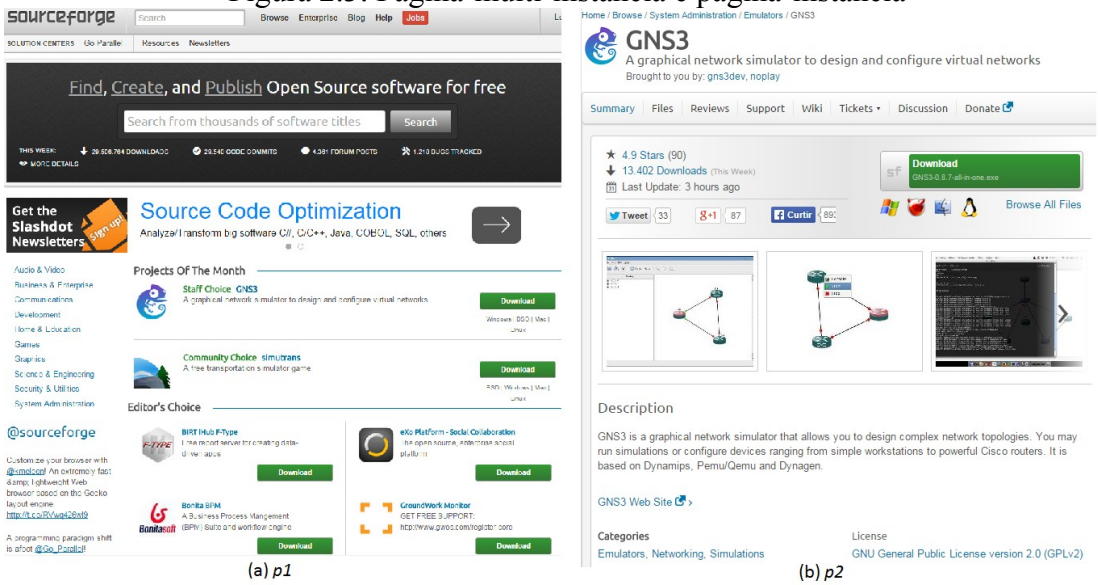
Os métodos de extração de dados da Web explorados neste trabalho se enquadram em dois tipos principais de redundância de conteúdo: a **redundância em nível de instância** (*instance level*), também denominada redundância em nível de entidade (*entity level*) e redundância em nível de objeto (*object level*), ocorre quando dois ou mais sites publicam um conjunto de instâncias em comum; e a **redundância em nível de atributo** (*attribute level*), também denominada redundância em nível de esquema (*schema level*) e redundância em nível de item de dados (*data-item level*), ocorre quando dois ou mais sites publicam um conjunto de atributos em comum.

As páginas que publicam dados de instâncias na Web podem ser classificadas quanto ao número de instâncias publicadas de uma determinada entidade. Uma **página-instância**, como definido por Blanco et al. (2008), é uma página que publica dados que representam uma única instância de uma determinada entidade. Uma **página-multi-instância** publica dados referentes a várias instâncias de

[5]http://pgfoundry.org/

uma determinada entidade. Por exemplo a Figura 2.3 mostra uma página-multi-instância (*p1*), que publica dados de vários projetos de software e uma página-instância (*p2*), que publica dados de um único projeto de software.

Figura 2.3: Página-multi-instância e página-instância



(a) *p1* (b) *p2*

Fonte: http://sourceforge.net/

As páginas que publicam dados de instâncias na Web podem ainda ser classificadas em duas subcategorias quanto ao uso de *template*, de acordo com Chang et al. (2006): **páginas baseadas em *template*** (*template-based pages*) e **páginas sem *template*** (*non-template pages*). Uma página baseada em *template* é gerada através do preenchimento de *templates* HTML fixos com dados geralmente armazenados em um banco de dados. Por exemplo, as páginas-instância de um projeto de software são geradas de acordo com o mesmo *template*. Uma página sem *template* é uma página sem um padrão, geralmente criada manualmente. Por exemplo, as listas de publicações em páginas pessoais de pesquisadores, onde cada publicação tem o título e o veículo de publicação, embora as páginas sejam produzidas de acordo com padrões diferentes.

DOM (*Document Object Model* – Modelo de Objeto de Documento), de acordo com a W3C[6], é uma interface de programação de aplicações (API) usada para definir a estrutura lógica de documentos (HTML ou XML) e como ele é acessado e manipulado. **Árvore DOM** é o nome da representação de uma página HTML em uma estrutura de árvore onde cada nodo é uma *tag* HTML (*body* ou *table*, por exemplo). Na página HTML, o elemento raiz (*/html*, por exemplo) e as *tags* de

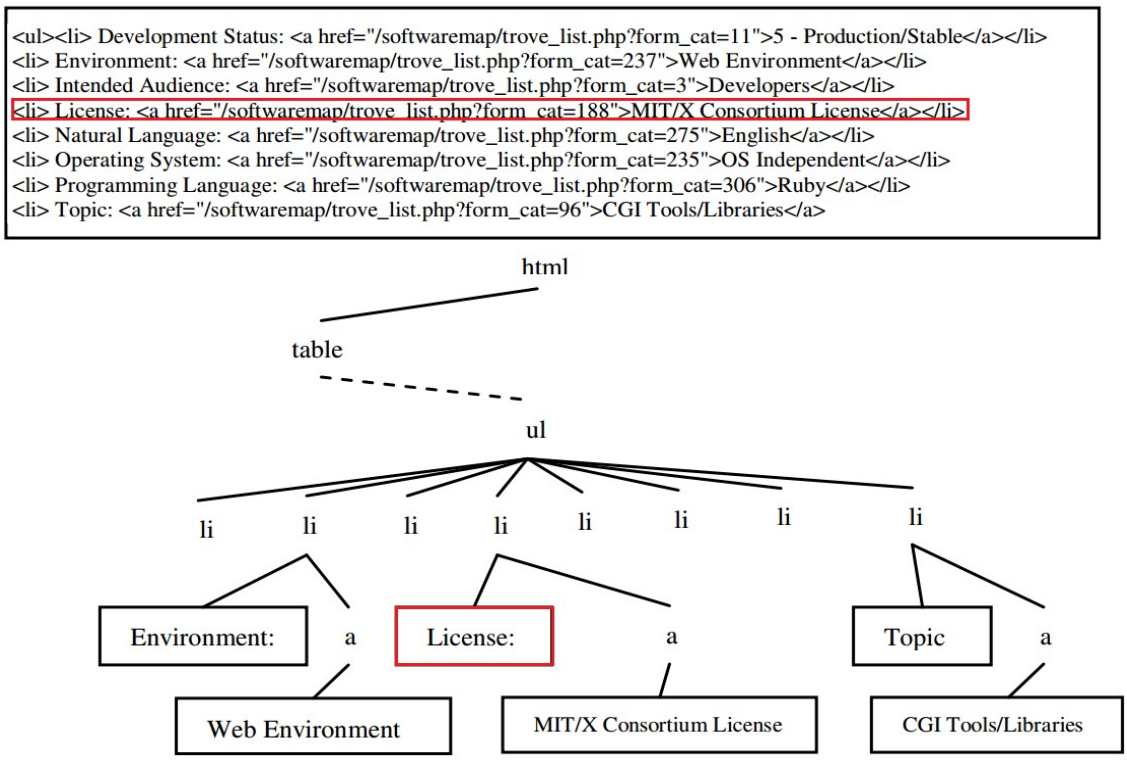
[6] http://www.w3.org/TR/DOM-Level-2-Core/introduction.html

texto (*/text*, por exemplo) são mapeados para o nodo raiz e os nodos folha – também chamados de nodos textuais – da árvore DOM, respectivamente. Por exemplo, a Figura 2.4 apresenta um exemplo de código HTML e sua árvore DOM correspondente.

O **caminho** de um nodo *x* é uma expressão XPath a partir de um determinado nodo (por exemplo, o nodo raiz) até o nodo *x* na árvore DOM. Um caminho pode ser: (i) **com índice** – inclui a informação da posição do nodo em relação aos seus irmãos; ou (ii) **sem índice** – não inclui a informação da posição do nodo em relação aos seus irmãos. Por exemplo, ainda na Figura 2.4 (em vermelho), o caminho sem índice do nodo contendo "*License:*", é denotado por */html/table/ul/li/text()*, e para o mesmo nodo, o caminho com índice é */html[1]/table[1]/ul[1]/li[4]/text()*.

Dados dois caminhos com um nodo em comum entre eles, um **caminho relativo** é um caminho que, primeiramente vai desde um determinado nodo em direção ao nodo folha, volta até o nodo em comum e então retorna em direção ao outro nodo folha. A notação de caminho relativo usa as barras ("/" e "\") para indicar as direções opostas. Por exemplo, na Figura 2.4, o caminho relativo entre o nodo textual "*License:*" e o nodo "*MIT/X Consortium License*" é */html/table/ul/li/text()/a/li\text()*.

Figura 2.4: Código HTML e sua árvore DOM



Fonte: Li et al. (2012)

Dado um conjunto de páginas de um site e um caminho para um determinado atributo, o **suporte** é o número de vezes que esse caminho ocorre no conjunto de páginas, para um único atributo, e o **caminho de maior suporte** é o caminho que tem mais ocorrências dentre os outros caminhos do mesmo conjunto.

3 EXTRAÇÃO DE DADOS BASEADO NA REDUNDÂNCIA DE CONTEÚDO

Este capítulo apresenta os métodos de extração de dados da Web baseados em redundância de conteúdo. A Seção 3.1 introduz a extração de dados da Web. A seção 3.2 descreve o método AERD e a Seção 3.3 apresenta o método FindAttrPos.

3.1 Extração de dados da Web

A Web é uma fonte de informações sobre produtos, negócios, livros, etc. Essas informações podem ser usadas tanto para uso acadêmico quanto comercial, mas, para um melhor uso e manipulação, essas informações devem ser extraídas da Web e inseridas em um banco de dados onde poderão ser comparadas.

3.2 Attribute Extraction based on Redundancy Detection (AERD)

Esta seção apresenta a descrição do método AERD na Seção 3.2.1 e seus detalhes de implementação na Seção 3.2.2.

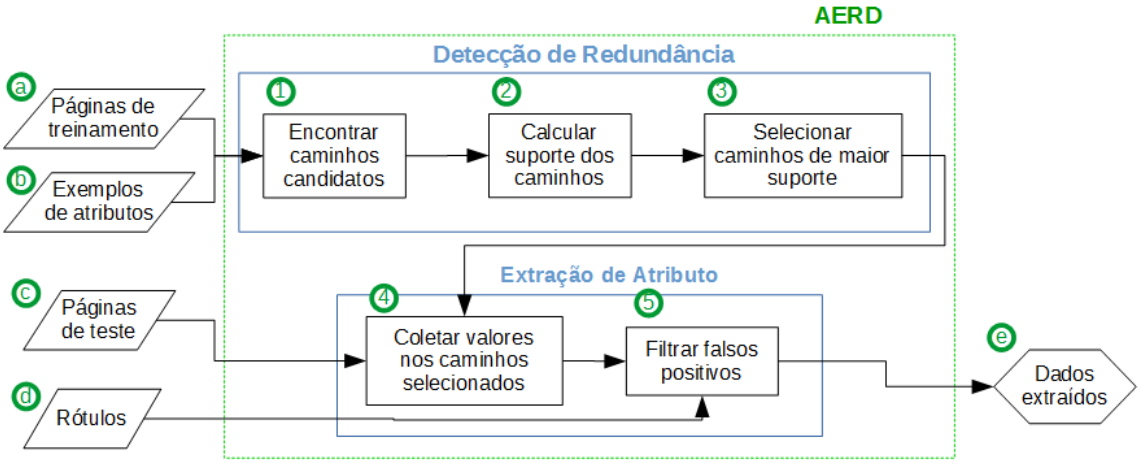
3.2.1 Descrição do método

Esta seção descreve a visão geral do método. Proposto por Li et al. (2012), o AERD (*Attribute Extraction based on Redundancy Detection* – Extração de atributos baseada na detecção de redundância), tem por objetivo extrair valores de atributos em um conjunto de páginas. O método é usado para extração de dados em sites de projetos de software de código aberto (*Open Source Forges*).

O AERD se baseia em um algoritmo de escaneamento por similaridade textual que recebe como entrada um conjunto de exemplos de atributos e um conjunto de páginas baseadas em *template* e, após uma etapa de detecção de redundância que analisa uma fração das páginas de entrada, aprende as posições dos atributos e as posições dos valores referentes a esses atributos. A saída é a extração de valores nas páginas de teste também inseridas como entrada. O AERD usa a notação de caminho sem índice para registrar as posições dos atributos nas páginas.

O método é dividido em duas etapas principais, como mostrado na Figura 3.1.

Figura 3.1: Fluxograma do método AERD



Fonte: Elaborado pelo autor.

A primeira etapa é a **detecção de redundância**, onde acontece o treinamento para encontrar os caminhos dos atributos. A primeira subetapa é **encontrar os caminhos candidatos** (1) de cada atributo nas **páginas de treinamento** (a) inseridas na entrada. Para se encontrar os caminhos, o AERD usa uma função que analisa cada nodo textual da página e calcula sua similaridade em relação aos **exemplos de atributos** (b), também inseridos na entrada. Caso a similaridade seja maior que um limiar, o caminho sem índice do nodo raiz até o nodo analisado é considerado um caminho candidato. A função de similaridade será detalhada mais adiante. A segunda subetapa é **calcular o suporte dos caminhos** (2) encontrados, ou seja, é contabilizado quantas vezes cada caminho encontrado ocorre nas páginas de treinamento. A última subetapa de detecção de redundância é **selecionar caminhos de maior suporte** (3): para um dado atributo, o caminho que tenha o maior número de ocorrências nas páginas fornecidas é considerado o caminho de maior suporte e selecionado para fazer as extrações na próxima etapa.

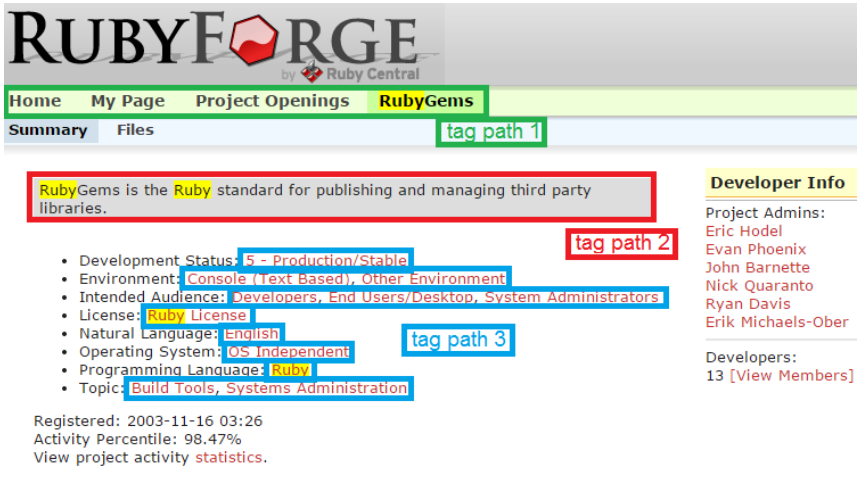
A segunda etapa do AERD é a **extração de atributo**. Nessa etapa ocorre a extração de valores dos atributos nas **páginas de teste** (c). A primeira subetapa é **coletar valores nos caminhos**

selecionados (4), ou seja, coletar valores nos caminhos de maior suporte[7]. Alguns valores podem ser coletados no mesmo caminho mas pertencerem a um rótulo diferente do desejado (essa situação ficará mais clara no exemplo a seguir) nesse caso o valor é considerado um falso positivo e é filtrado, usando os **rótulos** (d) como entrada, na última subetapa: **filtrar falsos positivos** (5). Após essa subetapa tem-se como saída os **dados extraídos** (e).

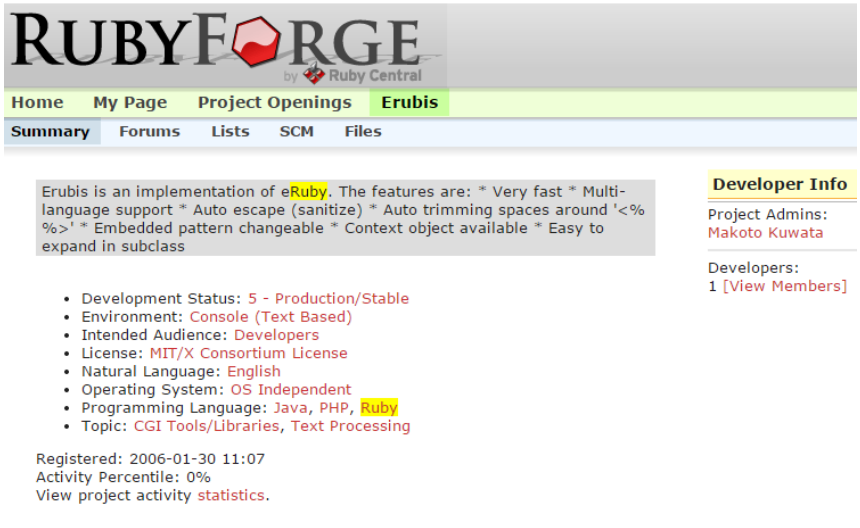
Para exemplificar a ideia do método são apresentadas três páginas (*p1*, *p2* e *p3*) baseadas em *template* do site *RubyForge*[8] na Figura 3.2. É possível verificar que na página *p1* o termo *Ruby,* que representa um valor do atributo linguagem de programação, ocorre em três caminhos diferentes (*tag path 1*, *tag path 2, tag path 3*).

[7]Número páginas que um caminho entrega valores similares aos valores de exemplos.
[8]https://**rubygems**.org/

Figura 3.2: Redundância nas páginas



(a) *p1*



(b) *p2*

(c) *p3*

Fonte: Elaborado pelo autor.

Esses caminhos estão descritos na Tabela 3.1 como o caminho do nodo raiz até o nodo textual que contém o termo. Para decidir qual é o caminho mais adequado, é verificado o suporte[9] de cada um. Escolhe-se a posição com o maior suporte. No exemplo, a posição *tag path 3* é escolhida para o atributo linguagem de programação, pois é a única que extrai valores desse atributo nas três páginas. Entretanto, outros 7 atributos estão sob o *tag path 3*. Dessa forma, são verificados os rótulos desses atributos para filtrar os falsos positivos, ou seja, os valores primeiramente coletados como corretos, mas, por não se localizarem exatamente sob o rótulo correspondente, são descartados como falsos. Por exemplo, o valor *OS Independent* é extraído por estar na *tag path 3* mas, como o nome do rótulo não é *Programming Language,* ele é excluído.

Tabela 3.1: Caminhos (*Tag Paths*)

	Tag path
1	/html/head/meta/title/text()
2	/html/body/div/table/tr/td/table/tr/td/p/text()
3	/html/body/div/table/tr/td/table/tr/td/ul/li/a/text()

Fonte: Li et al. (2012)

A função de similaridade utilizada é o Coeficiente de Jaccard baseado em q-gram. Essa função é definida como:

$$Similarity\,(s_i, s_j, q) = \frac{|qgram(s_i, q) \cap qgram(s_i, q)|}{|qgram(s_i, q) \cup qgram(s_i, q)|}$$

onde s_i e s_j são valores textuais e q é o coeficiente q-gram. Para cada valor textual, transforma-se cada letra em minúscula e a sentença resultante em um conjunto q-gram, por exemplo o conjunto 3-gram de "*Windows XP*" é {*##w, #wi, win, ind, ndo, dow, ows, ws#, s##, ##x, #xp, xp#, p##*}. Comparado com o conjunto gerado por "*Windoes XP*", que apresenta um erro no caractere "e", a função gera uma similaridade de 0,675 (5/8).

A principal contribuição à extração de conteúdo da Web do método AERD é o seu filtro de falsos positivos baseados na verificação dos rótulos dos atributos. Com o filtro é possível determinar se o valor a ser extraído pertence realmente ao atributo desejado.

A principal contribuição do método é, após a coleta de valores do site, filtrar os falsos positivos usando os rótulos para garantir que sejam referentes ao atributo desejado.

As desvantagens do método são a necessidade do uso de rótulo para sua execução e a falta de uma forma automática de descobrir os rótulos, transferindo esse trabalho ao usuário. Outra

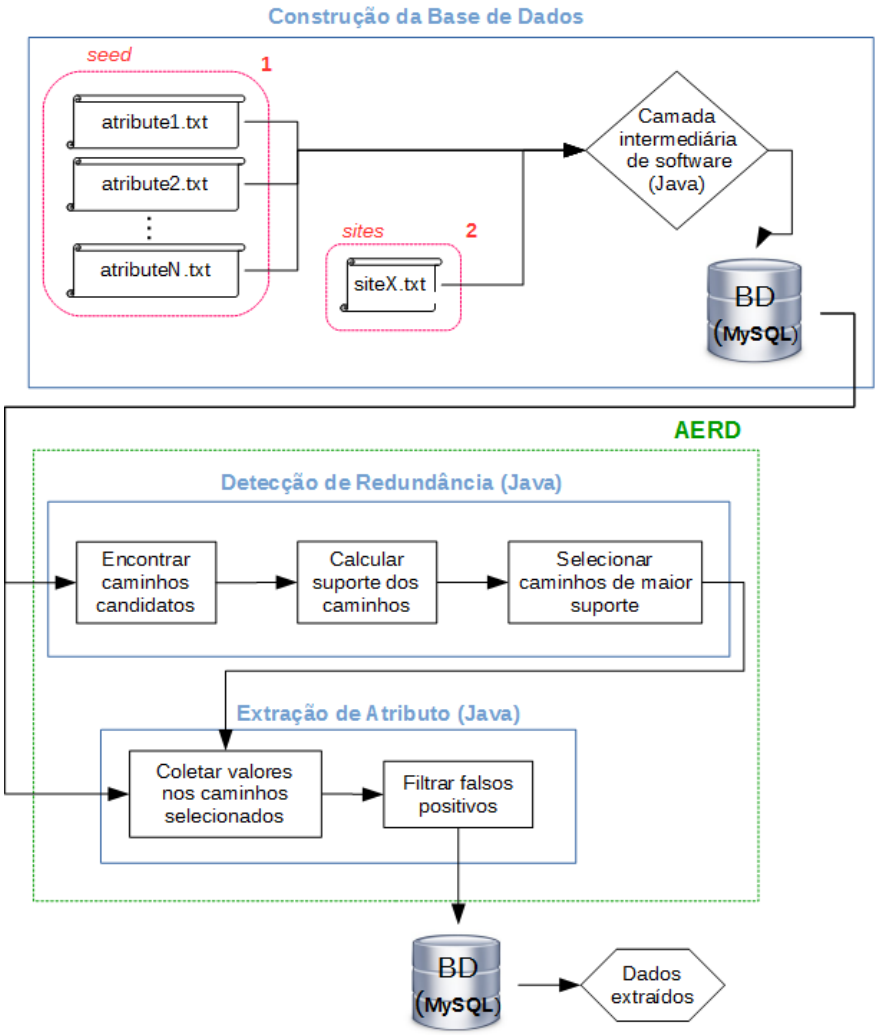
[9]Número páginas que um caminho entrega valores similares aos valores de exemplos.

desvantagem é a falta do uso de redundância em nível de instância, que traria mais robustez ao método na extração de valores referentes à instância que determinada página descreve.

3.2.2 Descrição da implementação

Esta seção descreve como foi implementado o método AERD. A implementação do autor, disponibilizada em *https://github.com/shockley/extractor*, requer como entrada um conjunto de exemplos (exemplo, Windows, Linux) para cada atributo (exemplo, sistema operacional) de cada entidade (exemplo, projeto de software). Além disso, para cada site devem ser fornecidos os rótulos dos atributos e os caminhos relativos. Entretanto, os caminhos relativos podem ser identificados automaticamente fornecendo uma página do site com os valores e rótulos anotados, isso acontece através da análise da página para localizar, através da função de similaridade, onde estão cada rótulo e valor, obtendo assim seus caminhos, os quais servirão de base para a etapa que analisa as páginas de treinamento. Essas informações, bem como a lista de páginas do site a serem extraídas, devem ser populadas em um banco de dados MySQL. No entanto, para facilitar a execução do método foi criada uma camada de software intermediária que lê essas informações de arquivos no formato CSV e popula o banco de dados. Dessa forma, antes de executar o método é necessário criar arquivos em duas pastas: *seed* e *sites*. A camada de software foi implementada pelo autor para a execução dos testes neste trabalho. O AERD foi implementado em Java.

Figura 3.3: Fluxograma da implementação do método AERD



Fonte: Elaborado pelo autor.

Na Figura 3.3 pode ser vista a ideia geral de como foi implementado o AERD. Na pasta ***seed*** **(1)**, deve ser criado um arquivo para cada atributo. Esse arquivo deve conter exemplos de valores do atributo. A Figura 3.4 apresenta um fragmento do arquivo com exemplos de valores para o atributo Categoria da entidade de projetos de software. Cada arquivo da pasta *seed* é único para cada entidade, ou seja, cada entidade possui seus valores de exemplo específicos para cada atributo, por exemplo, na entidade filmes, os exemplos devem ser dos atributos Diretor ou Gênero. A extração em diferentes sites de uma mesma entidade utilizam os mesmos arquivos desta pasta.

Figura 3.4: Arquivo de exemplos (*seed*) do atributo Categoria (*Category*) na entidade projetos de software

```
Adaptive Technologies
Artistic Software
Communications
BBS
Chat
AOL Instant Messenger
ICQ
Internet Relay Chat
Unix Talk
```

Fonte: Elaborado pelo autor.

Na pasta ***sites*** **(2)**, deve ser criado um arquivo para cada site. A Figura 3.5 apresenta um exemplo de arquivo. Primeiramente, se mapeiam os atributos que se deseja extrair, como Categoria (*Category*), Linguagem de Programação (*Programming Language*), entre outros, para os respectivos rótulos. Cada site possui páginas com os mesmos rótulos e sites diferentes precisam ter os rótulos mapeados para os atributos equivalentes. Após isso, são definidos manualmente os atributos e os respectivos valores de uma página de exemplo. Na Figura 3.5 pode-se observar os dados da página de exemplo S*kyTools*[10], cada entrada é constituída do endereço da página, o atributo e o valor correspondente, nessa ordem. O último passo é definir a lista de páginas que serão usadas para treinamento (páginas com o valor "1" ao final) e a lista de páginas que sera usados para testar o método (páginas com o valor "0" ao final).

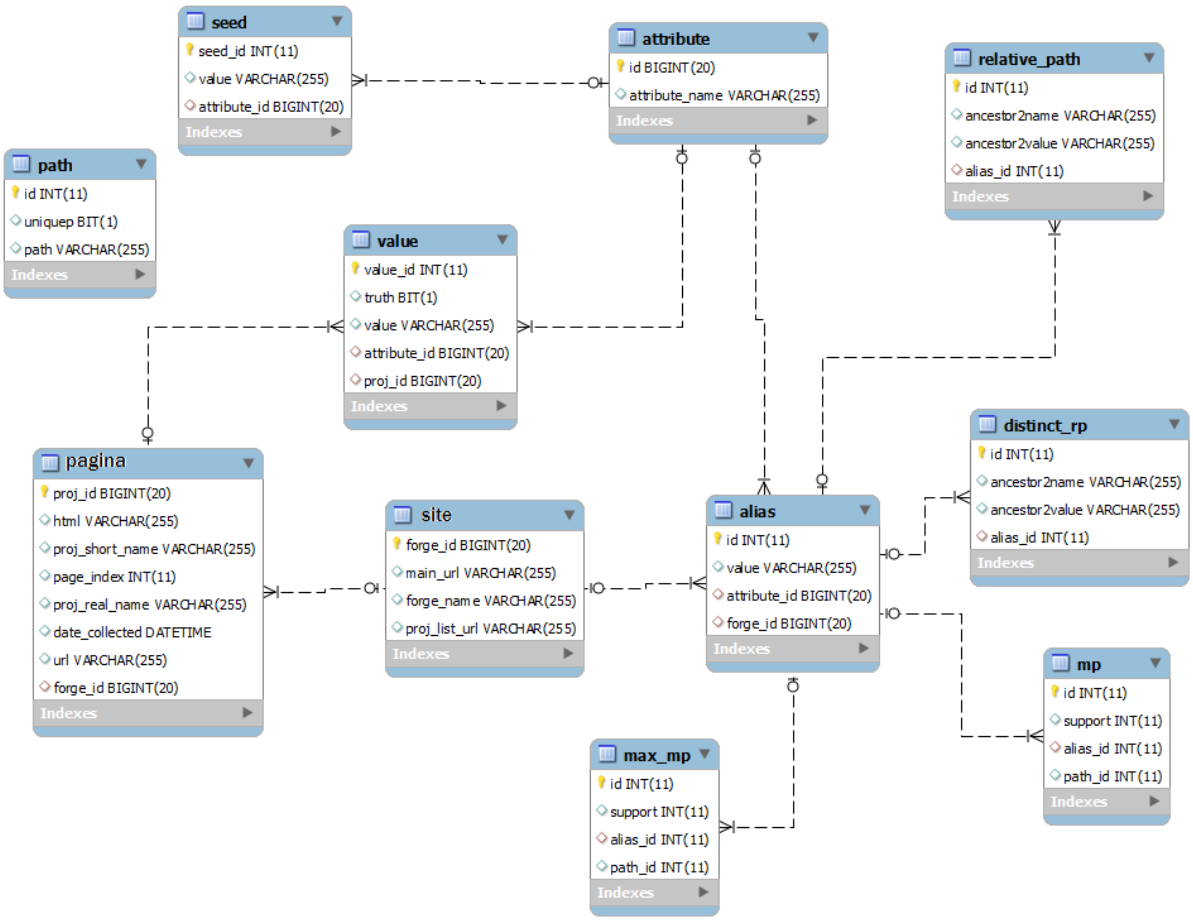
[10] http://pgfoundry.org/projects/skytools/

Figura 3.5: Arquivo de definição de páginas para extração

Fonte: Elaborado pelo autor.

A implementação do método utiliza o **banco de dados MySQL** para persistência dos dados. A Figura 3.6 mostra as tabelas do banco e seus relacionamentos. As principais tabelas armazenam as seguintes informações:

Figura 3.6: Tabelas do Banco de Dados

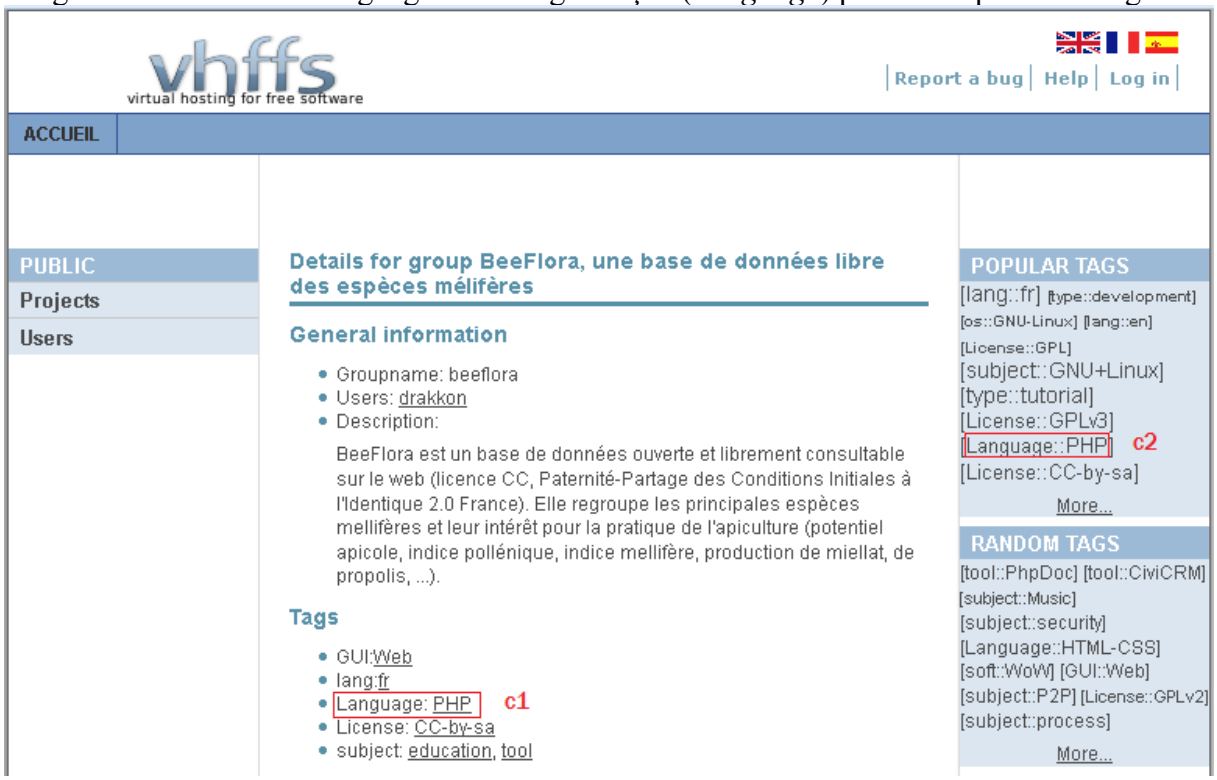


Fonte: Elaborado pelo autor.

- **seed –** exemplos de valores de cada atributo;
- **attribute –** lista de atributos aos quais serão extraídos os valores;
- **value –** lista dos valores extraídos;
- **alias** – rótulos dos atributos em cada site;
- **pagina –** informações de cada página coletada;
- **site –** informações dos sites, ou seja, o conjunto das páginas de uma dada extração (ex: PGFoundry, TuxFamily, Sourceforge)
- **relative_path -** armazena a informação do caminho relativo entre o nodo rótulo e o nodo valor de cada atributo em cada site. Podem ser armazenados caminhos incorretos, no caso de dois lugares contendo o mesmo rótulo como no exemplo da Figura 3.7, onde o caminho correto é *c1* mas o caminho *c2* pode ser visto erroneamente como válido pelo método;

- **distinct_rp** – registra as duplicatas da tabela *relative_path* quando esta possui caminhos incorretos para atributos;
- **mp** – guarda o cálculo do suporte de cada caminho relativo; e
- **max_mp** – armazena os caminhos relativos da tabela *mp* que possuem o maior suporte para cada atributo.

Figura 3.7: O atributo Linguagem de Programação (*Language*) pode ter 2 possíveis lugares



Fonte: *http://projects.tuxfamily.org/?do=group;name=beeflora*

O banco de dados deve estar vazio, ou seja, somente as tabelas, sem os valores, devem estar presentes antes da extração de uma entidade. As tabelas *attribute* e *seed* podem ser reutilizadas para a mesma entidade, ou seja, os mesmos valores podem ser usados para todos os sites da entidade projeto de software (ex.: PGFoundry, OW2, Tuxfamily). Outros valores devem ser utilizados para outra entidade.

Ao final da execução do treinamento e dos testes, os dados da extração são inseridos na tabela *value*, onde os valores podem ser mapeados para os seus respectivos atributos e para as páginas que geraram os dados.

3.3 Find Attribute Position (FindAttrPos)

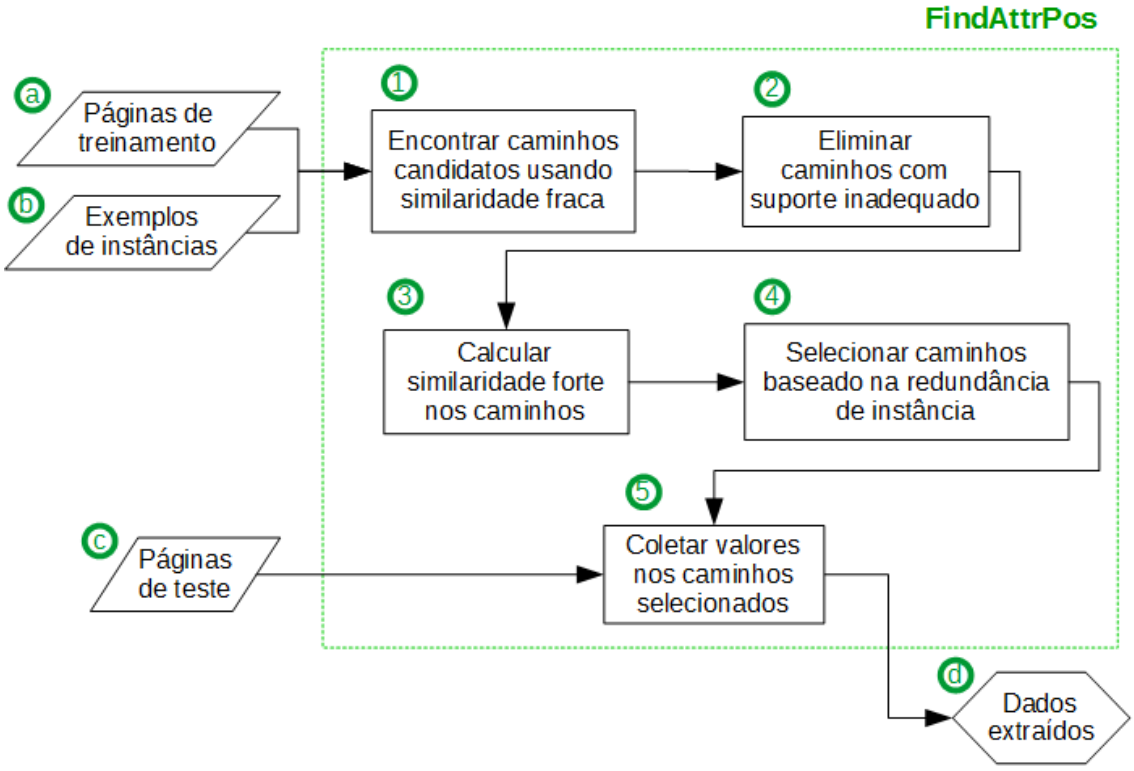
Esta seção apresenta a descrição do método FindAttrPos na Seção 3.3.1 e sua implementação na Seção 3.3.2.

3.3.1 Descrição do método

Esta seção descreve a visão geral do método. O FindAttrPos (*Find Attribute Position* – Encontre a posição do atributo), proposto por Gulhane et al. (2010), explora a redundância de conteúdo em nível de instância com o objetivo de extrair valores de atributos em um conjunto de páginas-instância baseadas em *template*.

O FindAttrPos é baseado em um algoritmo que realiza uma busca por similaridade textual entre valores em uma fração de páginas baseadas em *template* de um site, e exemplos de instâncias, ambos recebidos como entrada. Após algumas etapas de seleção dos melhores caminhos candidatos que contenham os valores dos atributos desejados, é selecionado somente um caminho para cada atributo. A saída do método é obtida do restante de páginas do site, onde são extraídos os valores dos caminhos de cada atributo.

Figura 3.8: Fluxograma do método FindAttrPos



Fonte: Elaborado pelo autor.

As etapas do método podem ser observadas na Figura 3.8. O FindAttrPos tem como entrada um conjunto de páginas-instância de um site. Uma fração dessas páginas são utilizadas para aprender os caminhos dos atributos no site e são chamadas de **páginas de treinamento (a)**. O restante de páginas do conjunto é usada para o teste de extração e são chamadas de **páginas de teste (c)**. O método também recebe como entrada um conjunto de **exemplos de instâncias (b)**, ou seja, exemplos de valores que ocorram nas instâncias de páginas. A saída do método são os **dados extraídos (d)** das páginas de teste.

A primeira etapa do FindAttrPos é **encontrar caminhos candidatos usando similaridade fraca (1)**, ou seja, através de uma função de similaridade, são procurados os caminhos nas páginas de treinamento com valores similares aos valores dos exemplos de instâncias. A função de similaridade fraca é uma variação de Gravano et al. (2003) que calcula a similaridade Cosseno usando *q-grams* em vez de palavras. A notação usada para expressar os caminhos é o caminho com índice a partir da raiz da árvore DOM. Por exemplo, o caminho com índice para o nodo textual de uma página que apareça na terceira *tag* do tipo *li* é: */html[1]/body[1]/td[1]/li[3]/text()*.

A segunda etapa é **eliminar caminhos com suporte inadequado (2)**, onde é contabilizado em quantas páginas um caminho entrega um valor similar aos valores de exemplo, ou seja, o suporte de cada caminho. Ao final dessa etapa são eliminados os caminhos que não atingem um número mínimo de suporte.

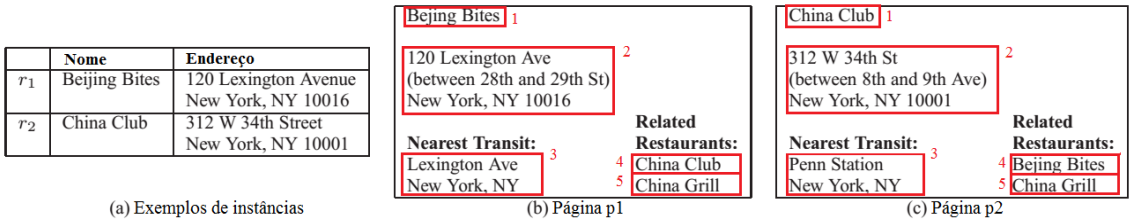
A terceira etapa é **calcular caminhos usando similaridade forte (3).** Nessa etapa, é empregada uma função de similaridade mais robusta, chamada similaridade forte. O objetivo dessa função é identificar padrões de casamento de segmentos textuais entre valores nas páginas de treinamento e valores nos exemplos de instância, e impulsionar os valores onde são identificados esses padrões. Após o cálculo dos novos escores de similaridade, os caminhos que não atingirem um escore mínimo são descartados.

A quarta etapa é **selecionar caminhos baseado na redundância de instância (5)**. Nela é escolhido, para cada atributo, um único caminho que: (i) extrai valores fortemente similares aos valores do atributo nos exemplos de instâncias em mais páginas de treinamento; e (ii) cada página que o caminho extrai valores fortemente similares também possui valores de outros atributos da mesma instância (redundância em nível de instância) extraídos por outros caminhos. Nessa etapa, um algoritmo baseado em Apriori (AGRAWAL; SRIKANT, 1994) é usado para podar caminhos de forma eficiente.

A última etapa é **coletar valores nos caminhos selecionados**, ou seja, após a seleção dos melhores caminhos, únicos para cada atributo, são obtidos os **dados extraídos (d)** desses caminhos. A extração de dados ocorre no restante de páginas-instância do site, que não foram usadas para treinamento, as **páginas de teste (c)**.

Para ilustrar a ideia do método são apresentados, na Figura 3.9, dois exemplos de instâncias, *r1* e *r2* **(a)** e duas páginas, *p1* **(b)** e *p2* **(c)**, com as posições de cada nodo textual assinaladas em vermelho.

Figura 3.9: Exemplos de instâncias e páginas web



Fonte: Gulhane et al. (2010)

Na primeira etapa do FindAttrPos, são procurados nas páginas *p1* e *p2*, valores fracamente similares aos valores dos registros *r1* e *r2*, para os atributos *Nome* e *Endereço*. Nessa etapa, são

encontrados quatro caminhos, um para cada uma das áreas **1**, **2**, **3** e **4**, em vermelho nas páginas *p1* e *p2* (os caminhos são idênticos para ambas as páginas já que elas tem a mesma estrutura). O Quadro 3.1 apresenta os escores de similaridade fraca e forte dos caminhos encontrados em relação ao registro *r1*. Do mesmo modo, o Quadro 3.2 apresenta os escores de similaridade em relação ao registro *r2.* Os escores de similaridade fraca encontrados para os quatro caminhos, em relação aos registros, são apresentados nas colunas 1, 3, 5 e 7 do quadro.

Quadro 3.1: Escores de similaridade fraca e forte para o registro *r1*

r1	**Página *p1***				**Página *p2***			
	Nome		**Endereço**		**Nome**		**Endereço**	
Similaridade	**Fraca**	**Forte**	**Fraca**	**Forte**	**Fraca**	**Forte**	**Fraca**	**Forte**
Caminho 1	0,9	0,9	0,0	0,0	0,0	0,0	0,0	0,0
Caminho 2	0,0	0,0	0,7	0,9	0,0	0,0	0,4	0,4
Caminho 3	0,0	0,0	0,6	0,6	0,0	0,0	0,3	0,3
Caminho 4	0,0	0,0	0,0	0,0	0,9	0,9	0,0	0,0
Caminho 5	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
Colunas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Fonte: Elaborado pelo autor.

Quadro 3.2: Escores de similaridade fraca e forte para o registro r2

r2	**Página *p1***				**Página *p2***			
	Nome		**Endereço**		**Nome**		**Endereço**	
Similaridade	**Fraca**	**Forte**	**Fraca**	**Forte**	**Fraca**	**Forte**	**Fraca**	**Forte**
Caminho 1	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0
Caminho 2	0,0	0,0	0,4	0,4	0,0	0,0	0,6	0,9
Caminho 3	0,0	0,0	0,2	0,2	0,0	0,0	0,7	0,7
Caminho 4	1,0	1,0	0,0	0,0	0,0	0,0	0,0	0,0
Caminho 5	0,6	0,6	0,0	0,0	0,5	0,5	0,0	0,0
Colunas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Fonte: Elaborado pelo autor.

Na segunda etapa, supondo um suporte[11] mínimo de $\beta = 1$, os quatro caminhos permanecem registrados, pois o suporte de cada um dos quatro caminhos é pelo menos β.

Na terceira etapa é calculada a similaridade forte para os valores dos caminhos. Os valores dos caminhos, ignorados os *templates* que influenciavam negativamente o escore de similaridade fraca (no caso os termos "*between 28th and 29th St*" em *p1,* e "*between 8th and 9th Ave*" em *p2*) tem seus escores de similaridade forte apresentados, para cada atributo, nas colunas 2, 4, 6 e 8 dos quadros. Ao final dessa etapa os caminhos que não atingirem um limiar $T_S = 0,9$, de similaridade forte, são descartados. Assim, permanecem os caminhos 1, 2, 4 e 5 para a próxima etapa.

Na quarta etapa são escolhidos os caminhos baseado na redundância de instância. Os caminhos 1 e 2 extraem valores fortemente similares aos valores dos registros. Cada página em que os caminhos 1 e 2 possuem valores fortemente similares aos valores dos registros também possui outros caminhos que extraem valores para a mesma instância, ou seja, para o mesmo registro. Por exemplo, o caminho 1 extrai um valor fortemente similar para *Nome* na página *p1* e também extrai um valor fortemente similar a *Endereço* na mesma página. Dessa forma, os caminhos 1 e 2 são escolhidos como únicos para extração de valores dos atributos *Nome* e *Endereço*, respectivamente, nas páginas do site.

A função de similaridade forte utilizada na terceira etapa do FindAttrPos leva em consideração a existência de *templates* nos valores de um atributo em um site. É verificado o padrão de casamento de segmentos entre os valores nos exemplos de instâncias e os valores de um caminho de atributo. O escore de similaridade entre dois valores é computado considerando apenas os segmentos que fazem parte do padrão de casamento, ignorando o *template*. Por exemplo, supondo um conjunto de exemplos de instâncias e um site sobre restaurantes com os atributos nome e endereço. Os exemplos de instâncias possuem os seguintes registros: *r1* = ("*Beijing Bites*", "*120 Lexington Avenue New York, NY 10016*"), *r2*= ("*China Club*", "*312 W 34th Street New York, NY 10001*"). O site possui os seguintes valores nas suas páginas: *p1* = ("*Beijing Bites*", "*120 Lexington Ave (**between 28th and 29th St**) New York, NY 10016*"), *p2* = ("*China Club", "312 W 34th St (**between 8th and 9th Ave**) New York, NY 10001*"). O site possui um *template* que ocorre em todas as páginas do site e não ocorre nos exemplos de instâncias (destacado). Esse *template* influencia negativamente o cálculo de similaridade fraca, pois não há um casamento nos exemplos de instâncias para o segmento do *template*. Com o cálculo da similaridade forte, os segmentos do *template* são ignorados das páginas

[11]Número páginas que um caminho entrega valores similares aos valores de exemplos.

e somente o restante, "*120 Lexington Ave New York, NY 10016*" em *p1* e "*312 W 34th St New York, NY 10001*" em *p2*, é comparado. Dessa forma, os termos das páginas casam, respectivamente, com os registros *r1* e *r2*, obtendo assim um escore de similaridade mais alto.

A principal contribuição do FindAttrPos é a definição da função de similaridade forte, mais robusta, que ignora frações de nodos textuais para encontrar valores referentes à instância que a página descreve.

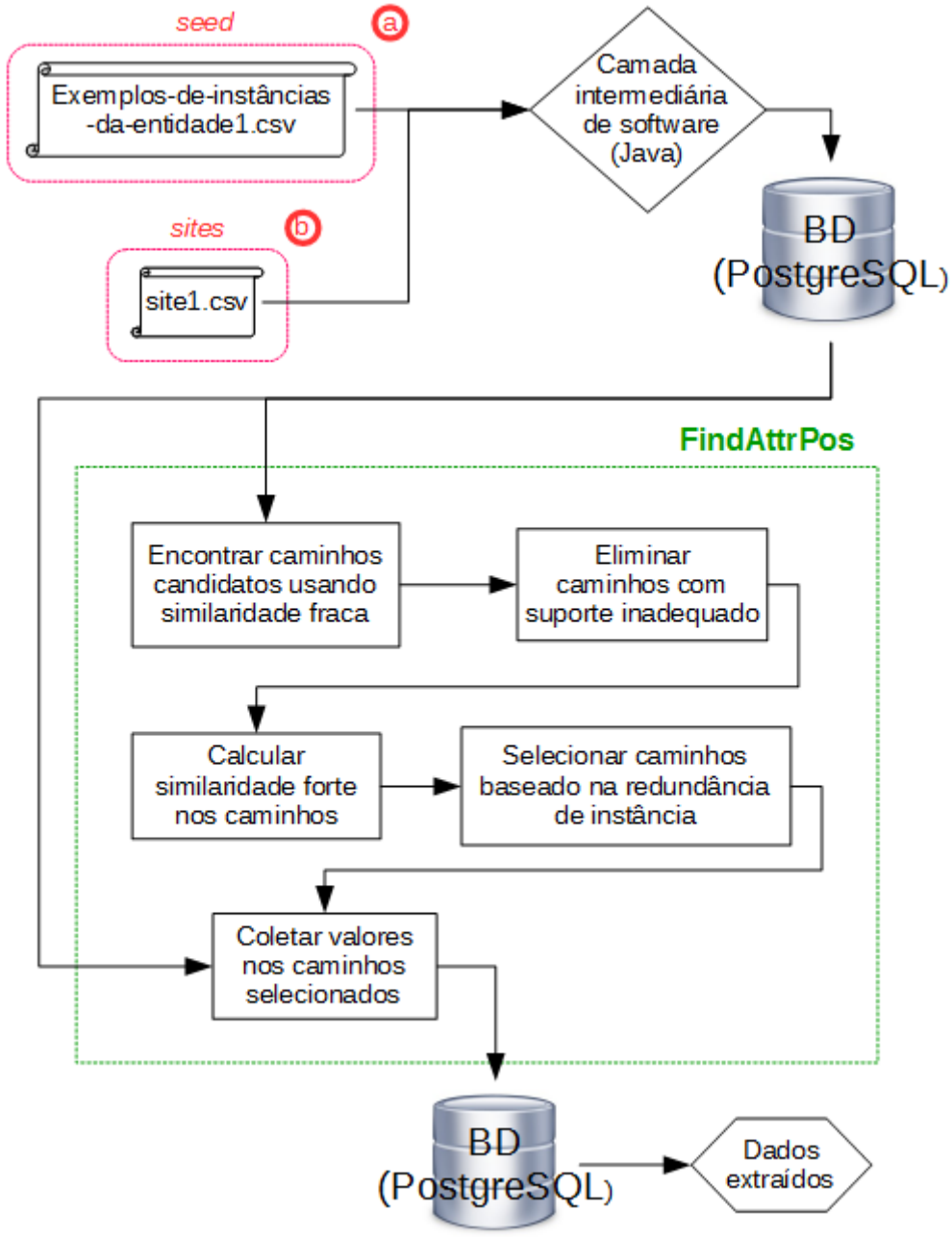
As desvantagens do método incluem a utilização de somente uma notação, caminho com índice, que não suporta a variação de posição dos atributos de um site. Outra desvantagem é a necessidade de inserir exemplos organizados por instância para garantir a redundância em nível de instância na extração de dados.

3.3.2 Descrição da implementação

Esta seção descreve como foi implementado o método FindAttrPos. Implementado como trabalho técnico por Michel et al. (2013), o método tem como entrada um conjunto de exemplos de instâncias, ou seja, um conjunto de valores de atributos anotados, e um conjunto de páginas de um site. Uma fração das páginas de entrada são usadas para treinamento e o restante é usado para teste de extração. O método procura nodos textuais similares aos valores dos exemplos de instâncias, nas páginas de treinamento, para identificar os caminhos ideais e realizar a extração de valores nesses caminhos. O FindAttrPos foi implementado em Java.

A Figura 3.10 apresenta o fluxograma da implementação do método FindAttrPos. A pasta ***seed* (a)** deve conter um arquivo, do tipo CSV, para cada entidade. Esse arquivo deve conter exemplos de instâncias de uma entidade. Na Figura 3.11, pode-se observar um arquivo com cinco exemplos de instâncias da entidade projeto de software. A primeira linha informa os atributos e a ordem que os valores devem ser anotados. Por exemplo, a primeira linha indica os atributos, em ordem, Categoria (*Topic*), Situação de Desenvolvimento (*Development Status*), Ambiente (*Environment*) e Público Alvo (*Intended Audience*). A segunda linha mostra os valores "*Database Administration*", "*5 – Production/Stable*", "*No Input/Output (Daemon)*" e "*Database Administrators*" respectivos aos atributos, na ordem em que aparecem.

Figura 3.10: Fluxograma da implementação do FindAttrPos



Fonte: Elaborado pelo autor.

Figura 3.11: Arquivo de exemplos de instâncias da entidade projeto de software

```
Topic,Development Status,Environment,Intended Audience
Database Administration,5 - Production/Stable,No Input/Output (Daemon),Database Administrators
Database Administration,3 - Alpha,Console (Text Based),Database Administrators
Database Development,4 - Beta,Win32 (MS Windows),Application Developers
Other/Non-listed Topic,5 - Production/Stable,Other Environment,Application Developers
Database Development,4 - Beta,NULO,Application Developers
```

Fonte: Elaborado pelo autor.

A pasta ***sites*** **(b)** deve conter um arquivo, do tipo CSV, para cada site. Na Figura 3.12, pode-se observar um exemplo de arquivo

com um conjunto de páginas do site *PGFoundry*[12]. As páginas podem ser referenciadas pelo endereço web ou, como no caso da figura, onde elas foram previamente baixadas, pelo caminho da unidade de disco rígido. Após a referência da página é preciso definir se a página deve ser usada para treinamento (com o valor "1" ao final) ou para teste de extração (com o valor "0" ao final).

Figura 3.12: Arquivo de páginas de site

```
pagina,treinamento
file:///C:\Sites\PGFoundry\web.archive.org\index.html,1
file:///C:\Sites\PGFoundry\web.archive.org\index-2.html,1
file:///C:\Sites\PGFoundry\web.archive.org\index-3.html,1
file:///C:\Sites\PGFoundry\web.archive.org\index-4.html,1
file:///C:\Sites\PGFoundry\web.archive.org\index-5.html,1
file:///C:\Sites\PGFoundry\web.archive.org\index-6.html,0
file:///C:\Sites\PGFoundry\web.archive.org\index-7.html,0
file:///C:\Sites\PGFoundry\web.archive.org\index-8.html,0
file:///C:\Sites\PGFoundry\web.archive.org\index-9.html,0
file:///C:\Sites\PGFoundry\web.archive.org\index-10.html,0
```

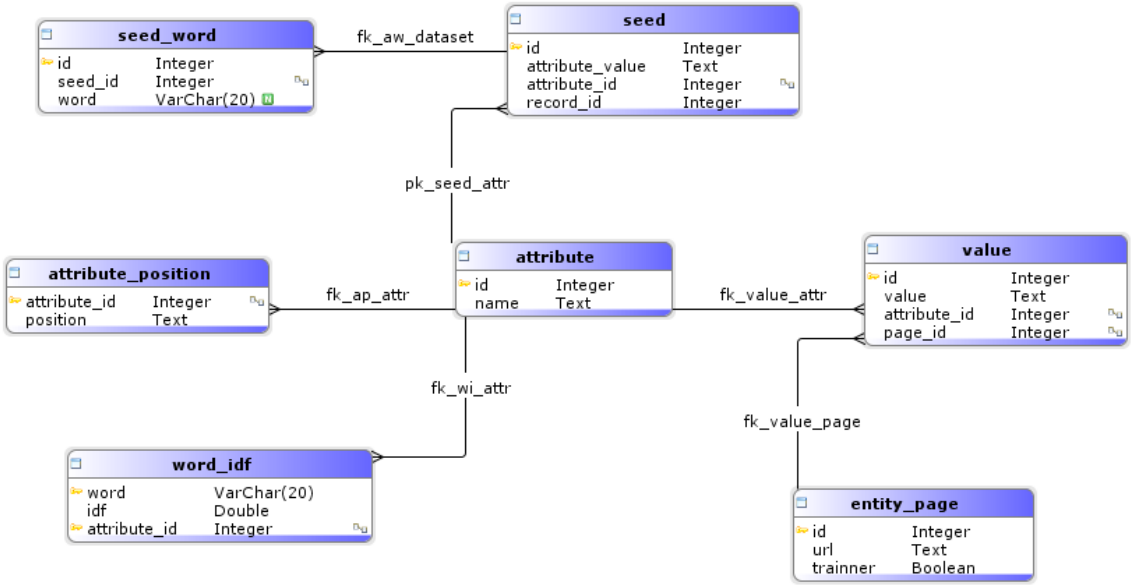
Páginas de treinamento

Páginas de teste

Fonte: Elaborado pelo autor.

O método foi implementado utilizando o banco de dados PostgreSQL para persistência dos dados. As tabelas e seus relacionamentos são apresentados na Figura 3.13. As tabelas armazenam as seguintes informações:

Figura 3.13: Tabelas do banco de dados PostgreSQL



Fonte: Elaborado pelo autor.

- **attribute** – lista dos atributos que deseja-se extrair;
- **attribute_position** – lista dos caminhos dos atributos no site;

[12] http://pgfoundry.org/

- **value** – lista dos valores extraídos das páginas;
- **entity_page** – informação de cada página usada;
- **seed** – lista de valores dos exemplos de instâncias;
- **seed_word –** lista das palavras, dos exemplos de instâncias, após um processo de tokenização. Essa lista é usada para comparar as palavras uma a uma no cálculo de similaridade forte;
- **word_idf** – peso de cada palavra usado para o cálculo da similaridade forte.

O banco de dados deve estar vazio, ou seja, somente as tabelas, sem os valores, devem estar presentes antes da execução do método. A camada intermediária de software da implementação do FindAttrPos se encarrega de limpar o banco de dados no início de cada execução.

Ao final da execução do treinamento e dos testes, os dados da extração são inseridos na tabela *value*, onde os valores podem ser mapeados para os seus respectivos atributos e para as páginas que geraram os dados.

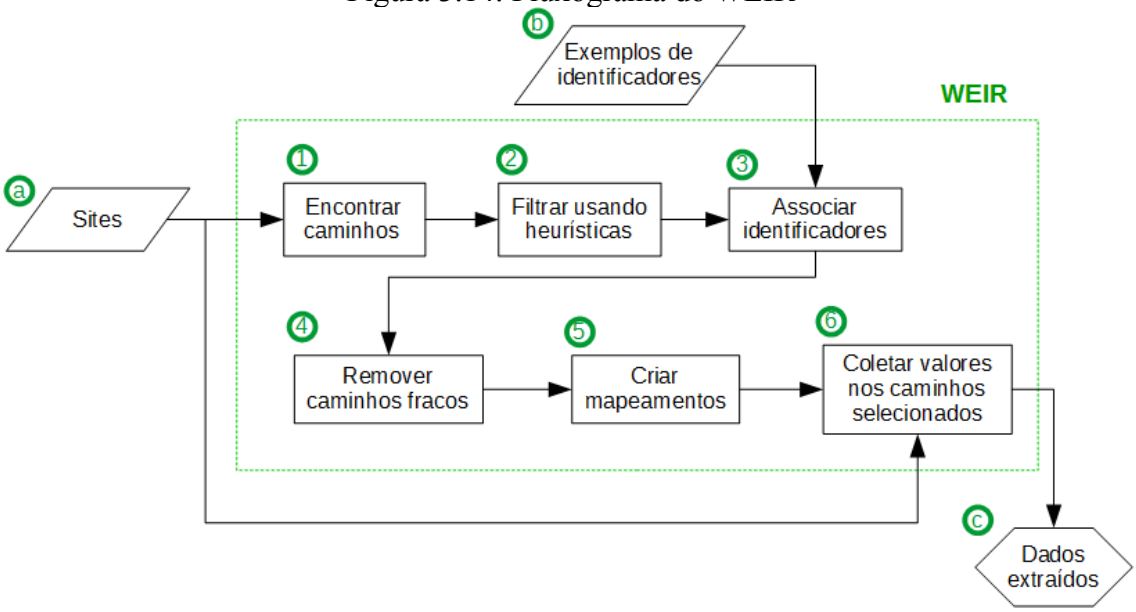
3.4 Web-Extraction and Integration of Redundant data (WEIR)

Esta seção descreve o método WEIR. O texto desta seção foi livremente adaptado de Manica (2015). O método WEIR (*Web-Extraction and Integration of Redundant data* – Extração e integração de dados redundantes na Web), proposto e implementado por Bronzi et al. (2013), tem como objetivo realizar a extração de dados em múltiplos sites baseado na redundância em nível de instância.

A Figura 3.14 apresenta as principais etapas do WEIR. A entrada é um conjunto de **sites (a)**, ou seja, de páginas-instância de diferentes sites com instâncias compartilhadas, e uma lista de **exemplos de identificadores (b)**. Um identificador é um atributo que identifica a instância descrita pela página. Para cada site, são gerados caminhos para extração. Esses caminhos são filtrados de acordo com heurísticas pré-definidas. Cada página é associada ao identificador da instância que ela descreve. Os caminhos são novamente filtrados a fim de remover caminhos fracos. Um caminho é considerado fraco se: (i) extrai os valores de um atributo em apenas uma fração das páginas do site; ou (ii) extrai valores de diferentes atributos nas páginas de um site. Finalmente, os caminhos de diferentes sites que extraem valores do mesmo atributo são agrupados (cada grupo é denominado

um mapeamento) e retornados. Os caminhos sem mapeamentos são descartados. A saída do método é um conjunto com os **dados extraídos (c)** dos sites inseridos na entrada.

Figura 3.14: Fluxograma do WEIR



Fonte: Elaborado pelo autor.

A primeira etapa, **encontrar caminhos (1)**, gera um conjunto de caminhos que ocorrem entre as páginas de um mesmo site. Essa etapa analisa a árvore DOM das páginas e classifica como nodos *template* os nodos textuais que ocorrem exatamente uma vez com o mesmo valor no mesmo caminho em um percentual significante (40%) de páginas. A notação usada para expressar os caminhos é o caminho sem índice a partir da raiz até o nodo textual, por exemplo, */html/body/span/li/text()*. Geralmente, o conteúdo textual de um nodo *template* é o rótulo de um atributo. Então, são gerados caminhos para os demais nodos textuais (nodos candidatos). Para cada nodo candidato, são gerados três tipos de caminhos: (i) um caminho com índice a partir do nodo raiz até o nodo candidato (exemplo, */html[1]/table[1]/tr[3]/td[2]/text()*); (ii) caminhos com índices a partir de nodos com um atributo *id* até o nodo candidato (exemplo, *//div[@**id**='tkr']/text()*); e (iii) caminhos com índices a partir de nodos *template* até o nodo candidato (exemplo, *//td[contains(text(),'Volume')]/../td[2]/text()*, supondo */td[contains(text(),'Volume')]* um nodo template). Para evitar a proliferação de caminhos a partir dos nodos *template*, são considerados apenas os caminhos com uma determinada distância (número de nodos) máxima entre o nodo *template* e o nodo candidato.

A segunda etapa, **filtrar usando heurísticas (2)**, aplica heurísticas para eliminar caminhos potencialmente desnecessários. São descartados os caminhos que não extraem valores em pelo menos 20% das páginas do site. Quando dois ou mais caminhos extraem exatamente os mesmos valores nas mesmas páginas, apenas o caminho com a maior preferência é selecionado e os demais são excluídos. Se os caminhos forem de tipos diferentes, tem a maior preferência de seleção o caminho a partir de um nodo *template* e a menor preferência de seleção o caminho a partir do nodo raiz. Se os caminhos forem do mesmo tipo, quanto menor a distância (número de nodos) maior a preferência de seleção do caminho.

A terceira etapa, **associar identificadores (3)**, encontra o identificador da instância que cada página descreve. Essa etapa se faz necessária, uma vez que as etapas seguintes exploram características da redundância em nível de instância, logo é necessário saber qual instância cada página descreve. Além dos caminhos gerados e filtrados nos passos anteriores, essa etapa requer como entrada um conjunto de identificadores de um pequeno número de instâncias presentes nos sites alvos da extração. Por exemplo, se a extração é realizada em sites sobre vereadores, um pequeno conjunto com nomes de vereadores deve ser fornecido. Em cada site, é encontrado o caminho com a maior interseção entre os valores extraídos e o conjunto de identificadores fornecido. O site que possui o caminho com maior interseção é selecionado. Os valores extraídos pelo caminho são associados como identificadores das páginas. Esses valores são utilizados para expandir o conjunto de identificadores fornecido e o processo é reiniciado para encontrar os identificadores nas páginas dos demais sites.

A quarta etapa, **remoção de caminhos fracos (4)**, processa todos os pares de caminhos do mesmo site que não possuem interseção entre seus valores ordenados pela distância não decrescente entre os seus valores. A distância é calculada através de uma nova função de distância baseada em tipo que compara apenas valores oriundos de páginas que descrevem a mesma instância (explorando a redundância em nível de instância). Para isso, é necessário que os sites compartilhem um número mínimo de instâncias. Assume-se que os pares de caminhos corretos são próximos e processados antes dos pares que incluem pelo menos um caminho fraco. Os caminhos de um par são marcados como corretos na primeira vez que são processados. Quando um par de caminhos é processado, se, pelo menos, um dos caminhos do par tem interseção não vazia com outro caminho do mesmo site marcado como correto, então esse caminho é considerado fraco e o par é ignorado. Finalmente, todos os caminhos não marcados como corretos são removidos.

A quinta etapa, **criar mapeamentos (5)**, tem como objetivo identificar caminhos de sites diferentes que extraem valores do mesmo atributo. Por exemplo, identificar caminhos que extraem valores do atributo partido em diferentes sites sobre vereadores. Inicialmente, cada caminho forma um mapeamento único. Em seguida, todos os pares de caminhos distintos são processados ordenados pela distância[13] não decrescente entre os seus valores. Se os dois caminhos do par são do mesmo site ou, pelo menos, um deles pertence a um mapeamento marcado como completo, então os mapeamentos que contém cada um dos caminhos do par são marcados como completos. Caso contrário, os mapeamentos que contém cada um dos caminhos são unidos. Essa estratégia assume que pares de caminhos de sites diferentes que extraem valores do mesmo atributo estão próximos e por isso são processados antes de pares de caminhos que extraem valores de atributos diferentes. Ao finalizar a criação dos mapeamentos, são eliminados os caminhos pertencentes aos mapeamentos únicos, ou seja, os caminhos que não possuem correspondentes em outros sites.

A última etapa é **coletar valores nos caminhos selecionados (6)**, ou seja, extrair os valores dos caminhos nos sites de entrada.

As vantagens do WEIR são: (i) eliminar a necessidade de exemplos ou de páginas previamente anotadas; e (ii) apresentar uma forma de descobrir o identificador da instância que cada página descreve.

Os principais problemas do WEIR são: (i) selecionar apenas um caminho por atributo, logo não suporta a variação de *template* para destacar os valores de um atributo em determinadas instâncias; e (ii) ser extremamente baseado na redundância de instância, sendo afetado quando os sites possuem períodos de tempo diferentes e os valores das instâncias mudam.

[13]A mesma função de distância baseada em tipo utilizada na etapa remover caminhos fracos (4).

4 AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta os experimentos realizados com o objetivo de comparar os métodos de extração de dados, AERD, FindAttrPos e WEIR, descritos no capítulo anterior, em termos de qualidade e eficiência. As bases de dados são apresentadas na Seção 4.1. A Seção 4.2 descreve os detalhes da configuração adotada nos experimentos. A Seção 4.3 define as métricas de avaliação. Os resultados são apresentados na Seção 4.4. Finalmente, a Seção 4.5 descreve os casos de falha.

4.1 Bases de dados

Esta seção detalha as bases de dados usadas na avaliação dos métodos de extração de dados. Foram utilizadas duas bases de dados: (i) projetos de software, com sites que possuem páginas-instância que descrevem projetos de software; e (ii) filmes, com sites que possuem páginas-instância que descrevem filmes.

A base de dados de **projetos de software** foi criada neste trabalho. Essa base inclui 5 sites. A Tabela 4.1 apresenta a descrição dos sites que compõem a base de dados de projetos de software. De cada site foram selecionadas 100 páginas-instância, cada página contendo em torno de 70 KB. O gabarito contendo o resultado esperado para a extração de dados em cada site foi criado manualmente, também neste trabalho, através da inspeção de páginas. Os atributos extraídos são: **Categoria**, **Situação de Desenvolvimento**, **Ambiente**, **Público Alvo**, **Licença**, **Plataforma** e **Linguagem de Programação**.

Tabela 4.1: Base de dados de projetos de software

site	descrição	projetos hospeados (aprox.)	endereço
OW2	A ObjectWeb 2 é uma comunidade de softwares *Open Source* para projetos colaborativos.	200	http://www.ow2.org/
PGFoundry	Comunidade colaborativa na web voltada para projetos que usem o Postgres.	380	http://pgfoundry.org/
FusionForge	Possui ferramentas para colaboração, fórum de mensagem e listas de e-mail.	1000	https://alioth.debian.org/
TuxFamily	Provê serviços sem custo para projetos que sigam a filosofia de Software Livre.	2600	http://www.tuxfamily.org/
SourceForge	Comunidade dedicada a ajudar projetos *Open Source*.	400000	http://sourceforge.net/

Fonte: Elaborado pelo autor.

A base de dados de **filmes** foi criada por Hao et al. (2011) e está disponível em http://swde.codeplex.com/, com o gabarito. Essa base de dados inclui 10 sites. A Tabela 4.2 apresenta os sites da base de filmes. Cada site possui 2000 páginas-instância, contendo em torno de 100KB cada uma. O gabarito é fornecido pelo site com a base. Os atributos extraídos são: **Diretor**, **Gênero**, **Classificação** e **Título**.

Tabela 4.2: Base de dados de filmes

site	descrição	endereço
Allmovie	Site de filmes. Se propõe a ser um recurso abrangente e aprofundado para saber mais sobre filmes, atores e cineastas.	http://www.allmovie.com/
AMCTV	Site sobre filmes e séries da produtora AMC.	http://movies.amctv.com/
Boxofficemojo	Site vinculado à IMDB. Publica dados de filmes e serviços de informação online.	http://boxofficemojo.com/
Hollywood	Site que publica comentários, notícias, trailers, fotos, vídeos e reportagens sobre filmes.	http://www.hollywood.com/
Iheartmovies	Rede social que permite aos utilizadores categorizar, avaliar e compartilhar suas coleções.	http://www.iheartmovies.com/
IMDB	Site que oferece uma base de dados pesquisável de filmes, programas de TV e de entretenimento.	http://www.imdb.com/
Metacritic	Site que publica a média ponderada de críticos que escrevem comentários on-line e impressos através do *Metascore*.	http://www.metacritic.com/
MSN	Site de filmes da Microsoft.	http://movies.msn.com/
Rottentomatoes	Site que publica dados da rede de parceiros, que incluem iTunes, Google, Target, DirecTV e Vudu.	http://www.rottentomatoes.com/
Yahoo	Site de filmes da Yahoo!	http://movies.yahoo.com/

Fonte: Elaborado pelo autor

4.2 Configuração de parâmetros

Os testes foram executados em um computador rodando Microsoft Windows 7. O computador tem a seguinte configuração de hardware: Processador Intel Core 2 Quad de 2,39 GHz, memória RAM de 3 GB e espaço de disco total de 150 GB.

A Tabela 4.3 detalha, para cada entidade, o número de páginas usadas para treinamento e teste em um único site, bem como o número de registros de exemplo para um atributo. Os métodos AERD e FindAttrPos usaram, para as bases de dados de projetos de software e de filmes, as mesmas configurações de quantidade de páginas de treinamento (páginas que foram usadas para cada método aprender as posições ideais para coletar valores) e de valores de exemplo (valores usados para o treinamento nas páginas). Os métodos usaram quantidades diferentes de páginas de teste (páginas em que ocorreu a extração de valores). Para a base de projetos de software foram usadas 100 páginas de cada site, dentre elas 20 foram selecionadas aleatoriamente para treinamento, e as 80 restantes foram usadas para realizar a extração. Os valores de exemplo foram coletados de 50 páginas de um único site (PGFoundry) e cada atributo recebeu no máximo 50 valores de exemplo. Para a base de filmes foram usadas 2000 páginas de cada site, dentre elas, 20 foram selecionadas aleatoriamente para treinamento, e 1980 foram usadas para a extração. Foram fornecidos 50 valores de exemplos de instâncias para cada atributo. Os valores de exemplo da base de filmes foram obtidos buscando uma intersecção entre os sites da base de dados, visto que, sem uma intersecção mínima, a extração retorna resultados com pouca qualidade.

Tabela 4.3: Quantidade de páginas de um único site usadas pelos métodos.

	total de págs	págs de treino (n)	págs de teste (n)	registros de exemplo (n)
Projetos de software	100	20	80	50
Filmes	2000	20	1980	50

Fonte: Elaborado pelo autor.

No método AERD é necessário fornecer rótulos dos atributos em cada site. A Tabela 4.4 apresenta os rótulos dos atributos em cada site da base de projetos de software. Os sites Tuxfamily e SourceForge não possuem os atributos Público Alvo e Sistema Operacional, respectivamente, e Situação de Desenvolvimento em ambos, dessa forma esses atributos (com valor "-" na tabela) não foram extraídos.

Tabela 4.4: Rótulos para atributos em sites de projetos de software

	Categoria	Situação de Desenvolvimento	Ambiente	Público Alvo	Licença	Sistema Operacional	Linguagem de Programação
PGFoundry	Topic	Development Status	Environment	Intended Audience	License	Operating System	Programming Language
OW2	Topic:	Development Status:	Environment:	Intended Audience:	License:	Operating System:	Programming Language:
TuxFamily	subject	-	GUI	-	License	os	Language
SourceForge	Categories	-	User Interface	Intended Audience	License	-	Programming Language
FusionForge	Topic	Development Status	Environment	Intended Audience	License	Operating System	Programming Language

Fonte: Elaborado pelo autor.

Os rótulos dos atributos nos sites de filmes para o AERD são mostrados na Tabela 4.5. Alguns atributos não possuem rótulos ideais nos sites. Por exemplo, o atributo Classificação no site Hollywood. Neste caso, foram escolhidos rótulos alternativos que estavam próximos ao valor do atributo, ou seja, os que possuem o caminho da árvore DOM mais semelhante ao caminho dos valores a serem coletados.

Tabela 4.5: Rótulos para atributos em sites de filmes para o AERD

	Diretor	Gênero	Classificação	Título
Allmovie	Director	Genres	MPAA Rating	Send to Friend
AMCTV	Director:	Genre/Type:	MPAA Rating:	details
Boxofficemojo	Director:	Genre:	MPAA Rating:	Domestic Total Gross:
Hollywood	Director	2 hr 0 mins	Create your own Fan Site on Hollywood.com. Click here!	Movies
Iheartmovies	Directed by	Genres	MPAA Rating	Film
IMDB	Director:	Genres:	Motion Picture Rating (MPAA)	More at IMDbPro
Metacritic	Director:	Genre(s):	Rating:	Studio:
MSN	Directed By:	Genre:	Rated:	On DVD
Rottentomatoes	Directed By:	Genre:	Rated:	On DVD
Yahoo	Directed by:	Genres:	MPAA Rating:	Production Photos

Fonte: Elaborado pelo autor.

As configurações do WEIR foram reaproveitadas de Bronzi et al. (2013). O WEIR foi usado para comparar resultados somente na base de filmes. O autor não disponibilizou uma implementação para realizar os testes na base de dados de projetos de software.

4.3 Métricas

Os métodos de extração foram avaliados nos aspectos de qualidade e eficiência. As métricas utilizadas para avaliar a qualidade dos resultados foram revocação, precisão e F1. A métrica utilizada para avaliar a eficiência foi o tempo de processamento.

A avaliação da qualidade tem como foco a capacidade dos métodos em extrair os valores corretos nas páginas. Um valor é considerado correto para um atributo A em uma página P, se ele representa o valor do atributo A para a instância I descrita pela página P.

A **revocação** mede o percentual de valores corretos extraídos por um método em relação ao total de valores corretos presentes em um site (BAEZA-YATES; RIBEIRO-NETO, 2011).

A **precisão** mede o percentual de valores corretos extraídos por um método em relação ao total de valores extraídos pelo método em um site (BAEZA-YATES; RIBEIRO-NETO, 2011).

A revocação e a precisão são definidas como seguem:

$$revocação = \frac{|Cor \cap Ext|}{|Cor|}$$

$$precisão = \frac{|Cor \cap Ext|}{|Ext|}$$

onde *Cor* é o conjunto de valores corretos, *Ext* é o conjunto dos valores extraídos por um método de extração e $|S|$ denota o número de elementos em um conjunto S.

F1 é a média ponderada harmônica entre precisão e revocação que atribui o mesmo peso para as duas métricas. F1 foi calculada como segue:

$$F1 = 2 \cdot \frac{precisão \cdot revocação}{precisão + revocação}$$

O **tempo de processamento** indica o tempo total gasto na execução do método para extrair os valores dos atributos em um site. Para ambas as bases de dados, as páginas de cada site foram previamente baixadas a fim de evitar a interferência da rede de dados no tempo de processamento. As páginas dos sites de projetos de software foram baixadas através do software *HTTrack Website Copier*[14].

Os valores de revocação, precisão e F1 foram obtidos com a média dos valores dos atributos em cada site. Desse modo, os valores da base de dados de filmes foram obtidos com a média dos 40 (10 sites vezes 4 atributos) valores individuais para cada atributo. Os valores da base de dados de projetos de software foram calculados com a média dos 31 (5 sites vezes 7 atributos, menos 4 casos onde o site não possuía alguns atributos) valores individuais para cada atributo. Os valores de tempo de processamento foram obtidos a partir da média de tempo de cada site das respectivas bases de dados. Desse modo, o tempo de processamento da base de filmes foi calculado com a média dos valores de tempo de processamento dos 10 sites, e o valor de tempo de processamento da base de projetos de software foi calculado através da média de tempo dos 5 sites.

O teste de Wilcoxon pareado (SIEGEL; CASTELLAN, 1988) avalia se as médias de duas distribuições de valores são estatisticamente diferentes, ou seja, não ocorreram ao acaso. O limiar de significância estatística utilizado nos experimentos foi $\alpha = 0{,}05$. Quando o valor da probabilidade P retornado pelo teste de Wilcoxon pareado é menor que α, existe uma diferença significativa entre os desempenhos dos métodos analisados, ou seja, a diferença de desempenho não ocorreu ao acaso. Com relação à qualidade, o melhor resultado é do método com a maior média de distribuição. Com relação à eficiência, o melhor resultado é o do método com a menor média de distribuição. O teste

[14] https://www.httrack.com/

de Wilcoxon foi utilizado porque as amostras não apresentaram uma distribuição normal. O teste de Wilcoxon foi obtido através da ferramenta Action[15], integrada ao Microsoft Excel.

4.4 Resultados

Esta seção apresenta os resultados dos testes de extração. Os resultados são descritos para a métrica de revocação na Seção 4.4.1, precisão na Seção 4.4.2, F1 na Seção 4.4.3 e tempo de processamento na Seção 4.4.4.

4.4.1 Revocação

O objetivo deste experimento é avaliar a revocação dos métodos. No total de páginas de um site, quanto mais páginas tiverem seus valores extraídos corretamente por um método, maior será sua revocação. Os valores de revocação foram obtidos com o cálculo da média de todos os atributos em todos os sites da base.

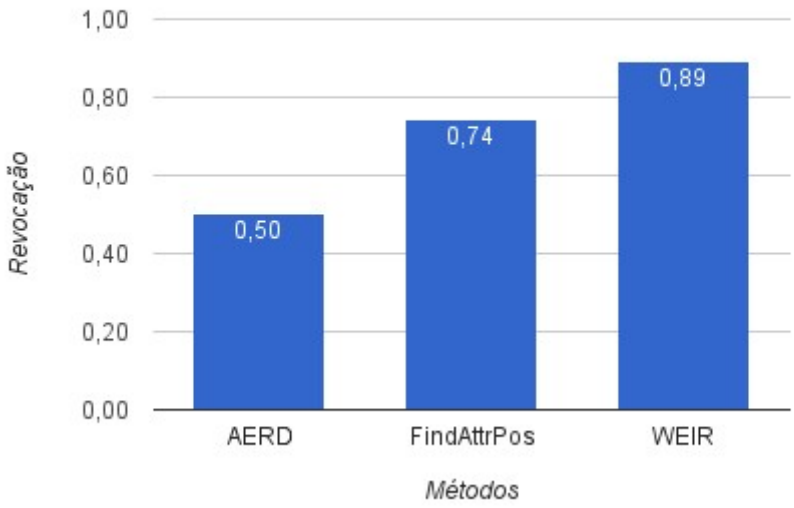
A Figura 4.1 apresenta os resultados de revocação média, para a base de dados de filmes, obtidos dos testes com os métodos AERD e FindAttrPos, assim como o resultado obtido de Bronzi et al. (2013), para o WEIR. Observa-se um resultado de 0,50 para o AERD, 0,74 para o FindAttrPos e 0,89 para o WEIR. O WEIR obteve um resultado mais alto por possuir um algoritmo mais robusto de redundância em nível de instância, que pode comparar valores de caminhos entre os sites e decidir o mais adequado para realizar a extração, além disso sua notação principal, caminho sem índice se comporta melhor nessa base. A base de dados de filmes possui poucos sites com atributos omissos e nenhum site com atributos com múltiplos rótulos em suas páginas, o que fez com que o FindAttrPos obtivesse um resultado próximo ao do WEIR. Uma característica que afetou especificamente o AERD foi a inexistência de um rótulo ideal para alguns atributos na maioria dos sites dessa base, fator essencial para esse método. Esse fato acarretou um resultado inferior aos outros métodos.

A diferença entre a revocação do AERD e do FindAttrPos, para a base de filmes, não é estatisticamente significante (valor de P do teste de Wilcoxon = 0,098)[16]. O valor P somente foi

[15] http://www.portalaction.com.br/
[16] Apêndice 1– Valores do Teste P de Wilcoxon para as métricas

calculado para os métodos AERD e FindAttrPos, já que não foram obtidos os valores de revocação individual do WEIR.

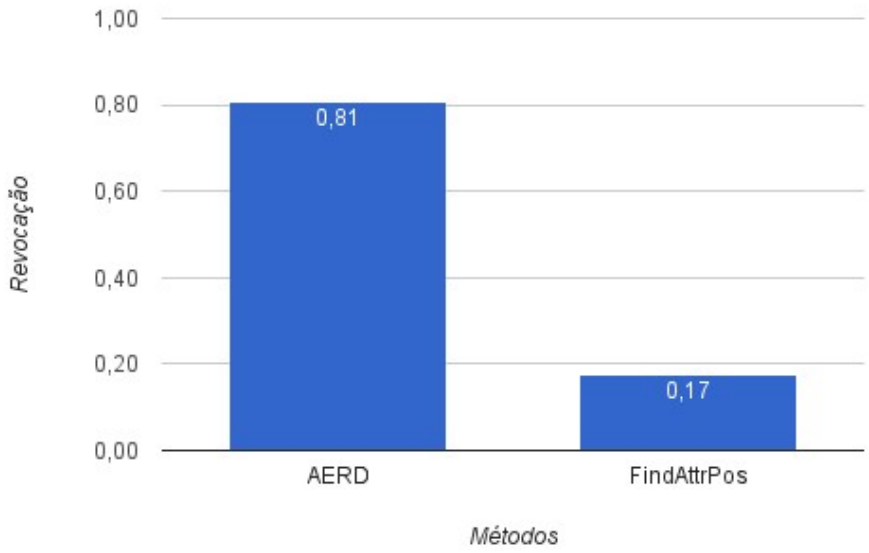
Figura 4.1: Revocação média na base de filmes



Fonte: Elaborado pelo autor.

Na base de dados de projetos de software foram obtidos os resultados de revocação média dos métodos AERD e FindAttrPos, implementados neste trabalho, apresentados no gráfico da Figura 4.2. Na figura, pode-se observar um resultado superior do AERD (0,81) em relação ao FindAttrPos (0,17). Uma das causas desses resultados, que afetou especificamente o FindAttrPos, se deve à possibilidade de atributos omissos ou com múltiplos rótulos para um mesmo atributo, em várias páginas dos sites dessa base de dados. Outra característica dessa base, que afetou principalmente o AERD, foi a existência, em caminhos diferentes de uma mesma página, de valores similares aos valores de exemplo dos métodos, competindo nas etapas de aprendizado dos mesmos. Os valores de revocação média do AERD e do FindAttrPos, para a base de projetos de software, não são estatisticamente significantes (valor de P do teste de Wilcoxon = 0,000027).

Figura 4.2: Revocação média na base de projetos de software



Fonte: Elaborado pelo autor.

4.4.2 Precisão

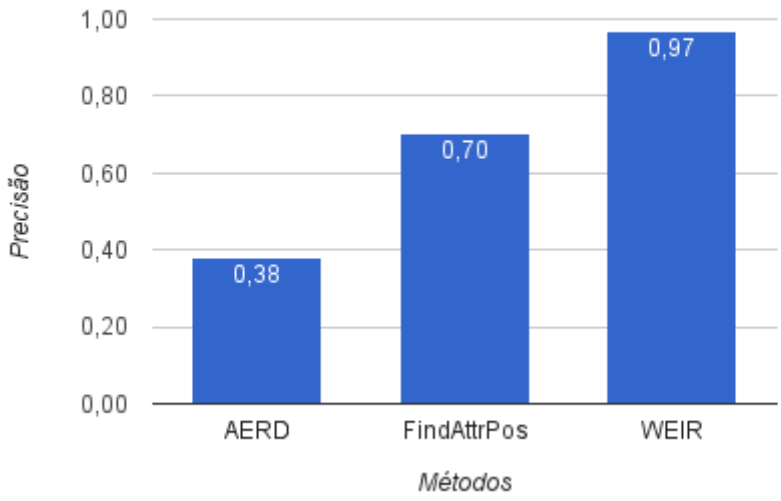
O objetivo deste experimento é avaliar a precisão dos métodos. Em relação ao total de valores extraídos por um método em um site, quanto mais valores forem corretos, maior será sua precisão. Os valores de precisão foram obtidos com o cálculo da média de precisão de todos os atributos das páginas nos respectivos sites.

A Figura 4.3 apresenta os resultados de precisão média na base de filmes obtidos nos experimentos dos métodos AERD e FindAttrPos, implementados neste trabalho, e o resultado do método WEIR, obtido de Bronzi et al. (2013). Na Figura 4.3, observa-se os valores de 0,38 para o AERD, 0,70 para o FindAttrPos, e 0,97 para o WEIR. A ocorrência de caminhos com valores referentes a instâncias diferentes à que a página descreve afetou negativamente ambos os métodos AERD e FindAttrPos. O WEIR se beneficiou da sua redundância em nível de instância mais robusta, comparando várias instâncias entre os sites para obter um caminho ideal de modo mais eficiente.

A diferença de precisão do AERD e do FindAttrPos, para a base de filmes, é estatisticamente significante (valor do teste P de Wilcoxon = 0,0076). O valor P somente foi calculado para os

métodos AERD e FindAttrPos, já que não foram obtidos os valores de precisão individual do WEIR.

Figura 4.3: Precisão média na base de dados de filmes

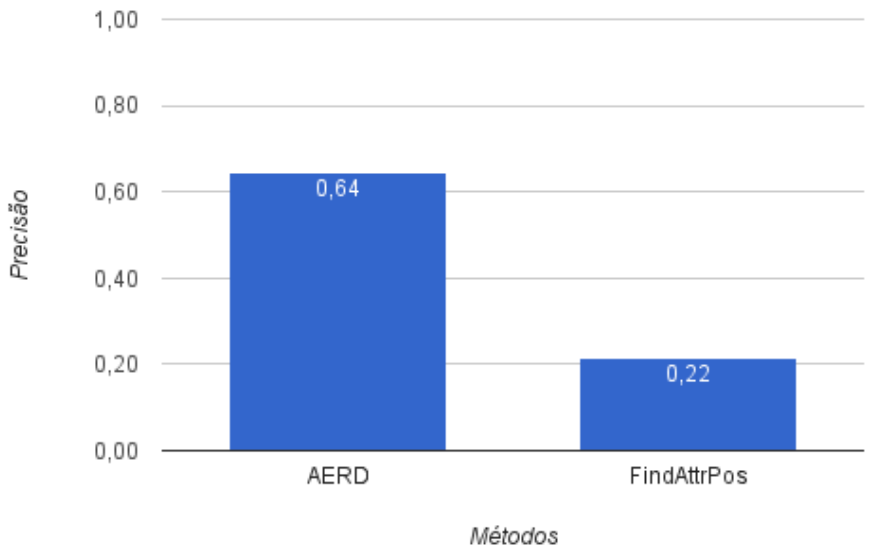


Fonte: Elaborado pelo autor.

A base de dados de projetos de software obteve os resultados de precisão apresentados no gráfico da Figura 4.4. Observa-se um resultado superior, de 0,64 para o AERD, seguido pelo resultado 0,22 para o FindAttrPos. O FindAttrPos foi prejudicado pela ocorrência de atributos omissos e existência de múltiplos rótulos de atributos, nas páginas da base de dados. Além disso, a ocorrência de caminhos com valores diferentes da instância que a página descreve, competindo com os valores referentes à própria instância que a página descreve, influenciou negativamente o principalmente o AERD.

A diferença de precisão média do AERD e do FindAttrPos, para a base de projetos de software, é estatisticamente significante (valor do teste P de Wilcoxon = 0,00046).

Figura 4.4: Precisão média na base de projetos de software



Fonte: Elaborado pelo autor.

4.4.3 F1

O objetivo deste experimento é avaliar a medida F1 dos métodos. A medida é um balanço entre a revocação e a precisão dos métodos. Os valores de F1 dos métodos AERD e FindAttrPos foram obtidos com o cálculo da média dos valores de F1 de todos os atributos da respectiva base de dados.

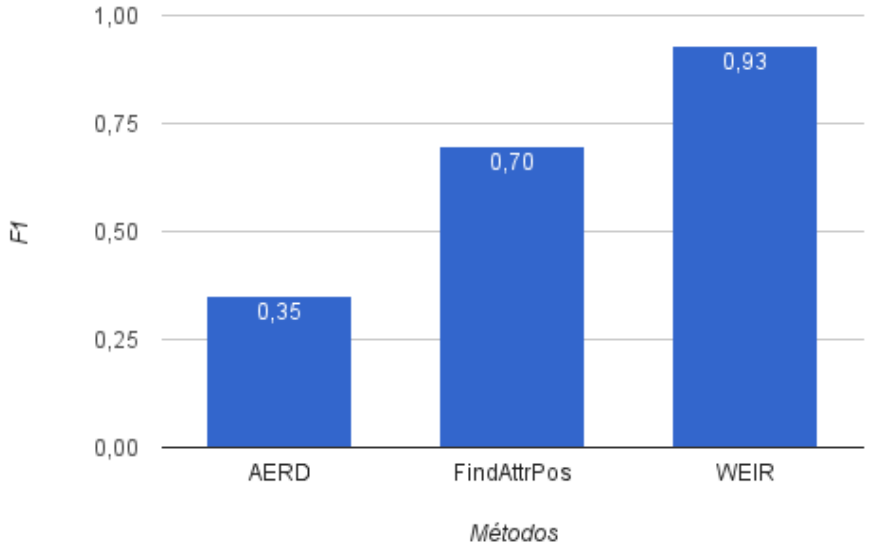
A Figura 4.5 apresenta os resultados de F1 média para a base de dados de filmes, obtidos dos testes com os métodos AERD e FindAttrPos, assim como o resultado obtido de Bronzi et al. (2013), para o WEIR.

Observa-se o resultado mais alto do WEIR (0,93), seguido do FindAttrPos (0,70) e do o AERD (0,35). Devido ao seu algoritmo mais robusto de redundância em nível de instância, o WEIR obteve um valor mais alto que os outros métodos. O FindAttrPos conseguiu uma relativa qualidade de extração devido à base de dados não possuir muitos sites com atributos omissos e nenhum site com múltiplos rótulos para um atributo. Essa característica é propícia para a notação do FindAttrPos, caminho com índice, que pressupõe que os atributos estejam sempre na mesma posição nas páginas

de um site. O AERD não obteve um valor de F1 alto nesta base pois necessita da informação de rótulo e a base possui a maioria dos sites com um atributo sem essa informação.

A diferença de F1 média do AERD e do FindAttrPos, para a base de filmes, é estatisticamente significante (valor do teste P de Wilcoxon = 0,0021). O valor P somente foi calculado para os métodos AERD e FindAttrPos, já que não foram obtidos os valores de F1 individual do WEIR.

Figura 4.5: F1 média na base de filmes

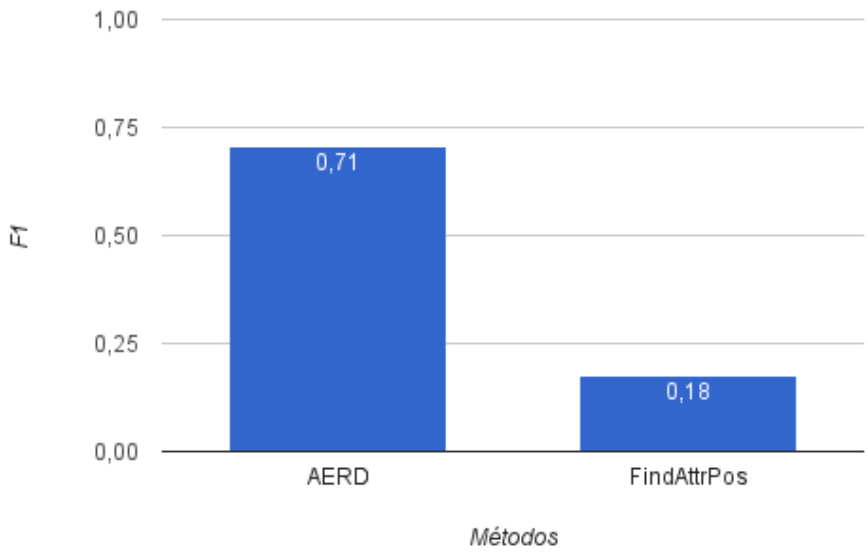


Fonte: Elaborado pelo autor.

A Figura 4.6 apresenta os valores de F1 média para a base de dados de projetos de software, obtidos dos testes com os métodos AERD e FindAttrPos, implementados neste trabalho. O AERD obteve o valor 0,71 e o FindAttrPos obteve 0,18. O FindAttrPos obteve um resultado menor pois utiliza a notação caminho com índice em seu algoritmo, que pressupõe que a posição dos atributos em um site não varie. A base de projetos de software possui atributos omissos e com múltiplos rótulos, o que faz com que a posição dos atributos varie entre uma página e outra, em um mesmo site. O AERD obteve um resultado superior pois suporta a variação de posição dos atributos, já que seu algoritmo não utiliza a informação de índice em sua notação.

A diferença de F1 média do AERD e do FindAttrPos, para a base de projetos de software, é estatisticamente significante (valor do teste P de Wilcoxon = 0,000056).

Figura 4.6: F1 média na base de projetos de softwares



Fonte: Elaborado pelo autor.

4.4.4 Tempo de processamento

O objetivo deste experimento é avaliar o tempo de processamento de uma execução dos métodos AERD e FindAttrPos, implementados neste trabalho. O tempo de processamento foi calculado com a média de tempo da execução de cada site de uma base de dados.

Para a base de dados de projetos de software foram obtidos os valores apresentados na Tabela 4.6. O AERD executou, em média, em 53 segundos enquanto o FindAttrPos executou, em média, em 27996 segundos (7 horas e 50 minutos, aproximadamente). O método FindAttrPos levou substancialmente mais tempo devido ao cálculo da similaridade forte, que analisa várias palavras de um nodo textual e procura um casamento com outros nodos textuais em toda uma página de um site. O tempo de processamento do FindAttrPos leva em consideração a quantidade de registros de exemplos de instâncias e de páginas de treinamento usadas no treinamento, fator limitante para os testes com o método, já que, em primeiros testes com cerca de 100 registros de exemplos de instâncias, uma execução levou mais de 10 dias, mesmo com poucas páginas de treinamento. Após as etapas de treinamento (cálculos de similaridade fraca, suporte, similaridade forte e redundância em nível de instância), o FindAttrPos levou poucos segundos para realizar a extração nos sites, visto

que ao final ele somente precisa coletar os valores nos caminhos escolhidos para a extração. O AERD precisou de menos de um minuto na maioria dos sites para realizar o cálculo de similaridade e alguns segundos para realizar a coleta de valores nos caminhos selecionados e filtrar os falsos positivos.

Tabela 4.6: Tempo de processamento na base de projetos de software

Projetos de software	**Tempo (segundos)**
AERD	53
FindAttrPos	27996

Fonte: Elaborado pelo autor.

Para a base de dados de filmes foram obtidos os valores de tempo de processamento apresentados na Tabela 4.7. O AERD levou, em média, 300 segundos (5 minutos, aproximadamente) enquanto o FindAttrPos levou 31038 (8 horas e 30 minutos, aproximadamente) para executar na base de filmes. No FindAttrPos, o cálculo do tempo de processamento na base de filmes, além dos fatores mencionados no cálculo de tempo da base de projetos de software, foi afetado pela quantidade de páginas de teste (1980 páginas) usadas em cada site. Apesar disso, o efeito é relativamente pequeno em relação à soma total de tempo, visto que a quantidade de páginas de teste só afeta a etapa de extração do método, e não as etapas de treinamento, que tomam tempo maior para serem executadas. O AERD foi afetado pelo número de página de teste, levando aproximadamente cinco vezes mais tempo para coletar os valores nas 1980 páginas da base de filmes em relação às 80 páginas da base de projetos de software. Ainda assim o tempo usado na extração foi relativamente menor ao FindAttrPos.

Tabela 4.7: Tempo de processamento na base de filmes

Filmes	**Tempo (segundos)**
AERD	300
FindAttrPos	31038

Fonte: Elaborado pelo autor.

4.5 Casos de falha

Esta seção apresenta os principais casos de falha dos métodos encontrados durante a avaliação experimental, as implicações na qualidade dos resultados dos métodos e os sites em que ocorreram. A seguir cada caso de falha é apresentado em uma subseção.

4.5.1 Atributos omissos

Atributos omissos ocorrem quando uma determinada instância não possui ou não foi publicado um valor, mesmo quando nulo (exemplo, "Categoria: <*NULL*>"), para um determinado atributo em uma página. Por exemplo, a Figura 4.7 apresenta duas páginas sobre projetos de software do site PGFoundry[17]. Na página *p1* (a) o atributo Ambiente ("*Environment*"), em vermelho, é publicado enquanto que na página *p2* (b) o atributo está omisso. Usando a notação de caminho com índice e supondo que o caminho ideal para o atributo Ambiente seja o caminho da página *p1*, por exemplo, */html[1]/body[1]/div[1]/li[2]/text()*, ou seja a segunda *tag* do tipo *li* da página. Caso realizasse a extração na página *p2* (b), o mesmo caminho devolveria o valor do atributo Público Alvo ("*Intended Audience*"), em verde, por se encontrar na segunda *tag* do tipo *li* dessa página.

Figura 4.7: Páginas com atributos omissos

(a) Página *p1* (b) Página *p2*

Fonte: http://pgfoundry.org/

[17] http://pgfoundry.org/

A qualidade da extração do FindAttrPos foi prejudicada por esse caso, devido à notação utilizada para expressar as posições dos atributos, caminho com índice. Essa notação pressupõe que os atributos sempre ocorram na mesma posição em todas as páginas de um site. A revocação foi afetada, pois em alguns casos o método não foi capaz de extrair o valor correto por estar em uma posição diferente da aprendida. A precisão foi afetada, pois em alguns casos o método coletou valores incorretos, por estarem no caminho aprendido.

O caso de falha de atributos omissos ocorreu em todos os sites da base de dados de projetos de software. O caso de falha ocorreu em 4 sites (AMCTV, IMDB, Metacritic e Yahoo) da base de dados de filmes. Em muitos casos dessa base o valor não é publicado mas o atributo, com o respectivo rótulo, está presente, o que não interferiu nas posições dos atributos na página.

4.5.2 Atributo com múltiplos rótulos

Um atributo com múltiplos rótulos acontece quando um atributo de uma instância possui mais de um valor e é usado mais de um rótulo para publicar esse valor. Os atributos podem variar o número de rótulos usados de uma página para outra fazendo com que a posição em que os atributos aparecem varie em um site. Por exemplo, a Figura 4.7 apresenta um exemplo do caso de mais de um rótulo para o mesmo atributo. Na página *p1* (a), o atributo Público Alvo ("*Intended Audience*"), em vermelho, possui somente um rótulo enquanto que na página *p2* (b) o mesmo atributo possui 2 rótulos, em vermelho. O atributo Licença ("*License*") é alterado de terceira para quarta posição, respectivamente, nas páginas *p1* e *p2,* em verde.

Figura 4.7: Atributo com múltiplos rótulos

(a) Página *p1* (b) Página *p2*

Fonte: http://pgfoundry.org/

A qualidade da extração do FindAttrPos foi prejudicada. A notação de caminho com índice, utilizada pelo FindAttrPos, pressupõe que os atributos sempre ocorram na mesma posição em todas as páginas de um site. A revocação foi afetada, pois em alguns casos o método não foi capaz de extrair o valor correto por estar em uma posição diferente da aprendida. A precisão foi afetada, pois em alguns casos o método coletou valores incorretos, por estarem no caminho aprendido mas não pertencerem ao atributo desejado.

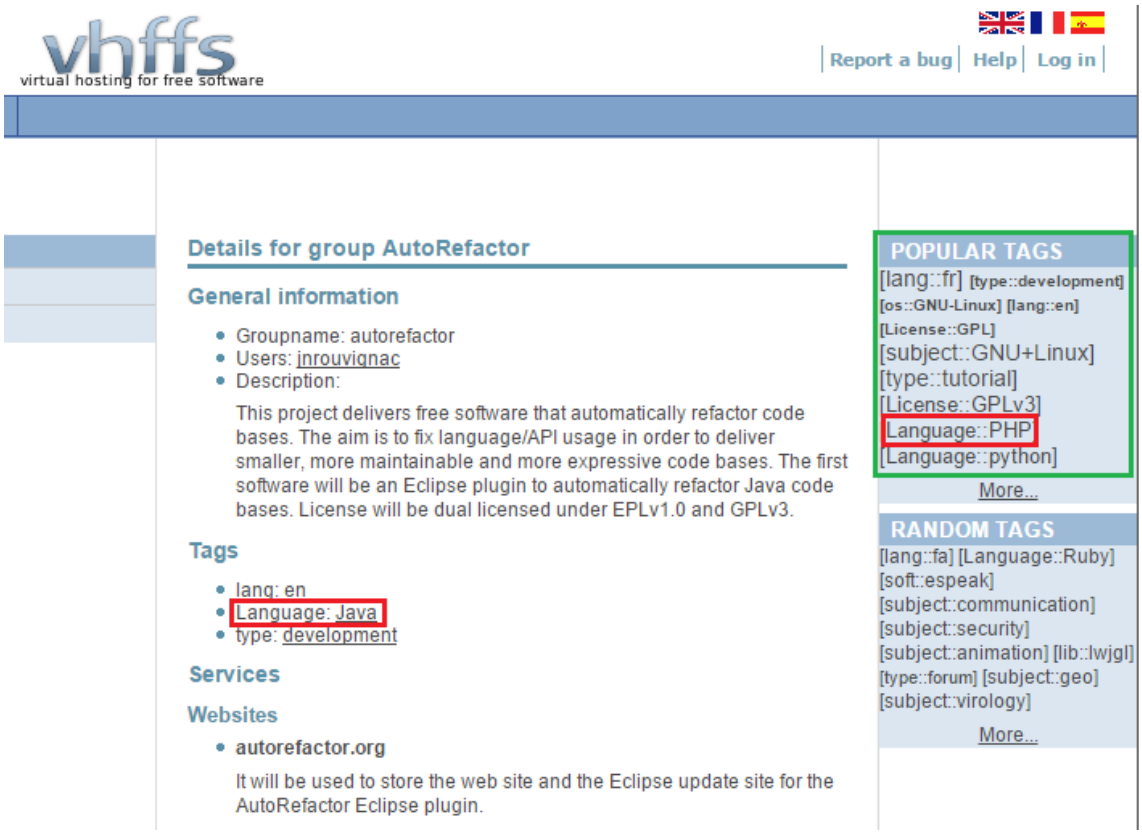
A base de dados de projetos de software possui atributos com múltiplos rótulos em 2 de seus sites (PGFoundry e Alioth). A base de filmes não possui múltiplos rótulos para um atributo, em vez disso, essa base publica mais de um valor, quando a instância possui, sob um mesmo rótulo.

4.5.3 Valores de instâncias diferentes à que a página descreve

Em determinados sites ocorrem valores do domínio de um atributo que não representam a instância descrita na página. Esses valores são similares aos valores de exemplos de entrada dos métodos e por isso são possíveis de terem seus caminhos considerados como ideais nas etapas de treinamento dos mesmos. Por exemplo, na Figura 4.8 pode-se observar uma página que descreve a instância do projeto de software *AutoRefactor*. A página apresenta, além do atributo Linguagem de Programação ("*Language*"), com o valor "*Java*", ao centro, em vermelho, da própria instância da

página, também o valor "*PHP*", à direita, em vermelho, também do atributo Linguagem de Programação mas de uma instância diversa. É possível haver atributos omissos no site *Tuxfamily*[18], para as instâncias que as páginas descrevem. O campo "*POPULAR TAGS", em verde,* onde ocorrem atributos de instâncias diversas, sempre ocorre em todas as páginas. O caminho do campo "*POPULAR TAGS"* compete com o caminho ideal, ou seja, o caminho que contém o valor do atributo Linguagem de Programação, da instância da página. Desse modo, o caminho incorreto tem maior suporte no total de páginas usadas para treinamento e é eventualmente escolhido para realizar a extração.

Figura 4.8: Página com atributos referentes a uma instância diversa à que a página descreve



Fonte: http://projects.tuxfamily.org/?do=group;name=autorefactor

O AERD foi prejudicado por esse caso. O método não compara se os valores da página pertencem a uma determinada instância. A revocação foi afetada, pois em alguns casos o método não foi capaz de extrair o valor correto, pois o mesmo não pertence ao atributo da instância que a

[18] *http://tuxfamily.org/*

página descreve. A precisão foi afetada, pois em alguns casos o método coletou valores incorretos, por estarem no caminho aprendido, mas não pertencerem à instância da página.

O caso de valores de instâncias diferentes à que a página descreve ocorreu em 1 site (Tuxfamily) da base de dados de projetos de software. O caso ocorreu em 7 sites (todos exceto Iheartmovies, Boxofficemojo e Yahoo) da base de filmes.

4.5.4 Atributo sem rótulo

Um atributo sem rótulo ocorre quando uma página publica o valor de um atributo mas esse atributo não possui um rótulo. Por exemplo, a Figura 4.9 apresenta uma página da base de dados de filmes. O valor "*Coal Miner's Daughter (1980)*", referente ao atributo Título, não possui um rótulo (por exemplo, "*Title:*").

Figura 4.9: Página sem rótulo para o atributo Título

Fonte: http://www.rottentomatoes.com/m/coal_miners_daughter/

A qualidade de extração do AERD foi prejudicada por esse caso de falha devido à necessidade do método usar rótulos para os atributos. O método utiliza os rótulos para verificar se o valor extraído pertence ao atributo desejado. No caso do rótulo não corresponder ao valor configurado para o site, ele é descartado. A revocação do AERD foi afetada pois o método não extraiu valores para os atributos que não possuíam rótulos. A precisão do AERD foi afetada, pois o método coletou valores incorretos nas páginas dos sites. Quando foi configurado um nodo textual próximo ao

atributo, na tentativa de suprir a necessidade do rótulo, o método coletou valores incorretos e não descartou devido ao rótulo também estar próximo do valor incorreto.

O caso de falha de atributo sem rótulo ocorreu em 9 (todos exceto Iheartmovies) sites da base de dados de filmes, somente para o atributo Título. O caso de falha não ocorreu na base de dados projetos de software.

4.5.5 Valores fora da intersecção entre exemplos e site

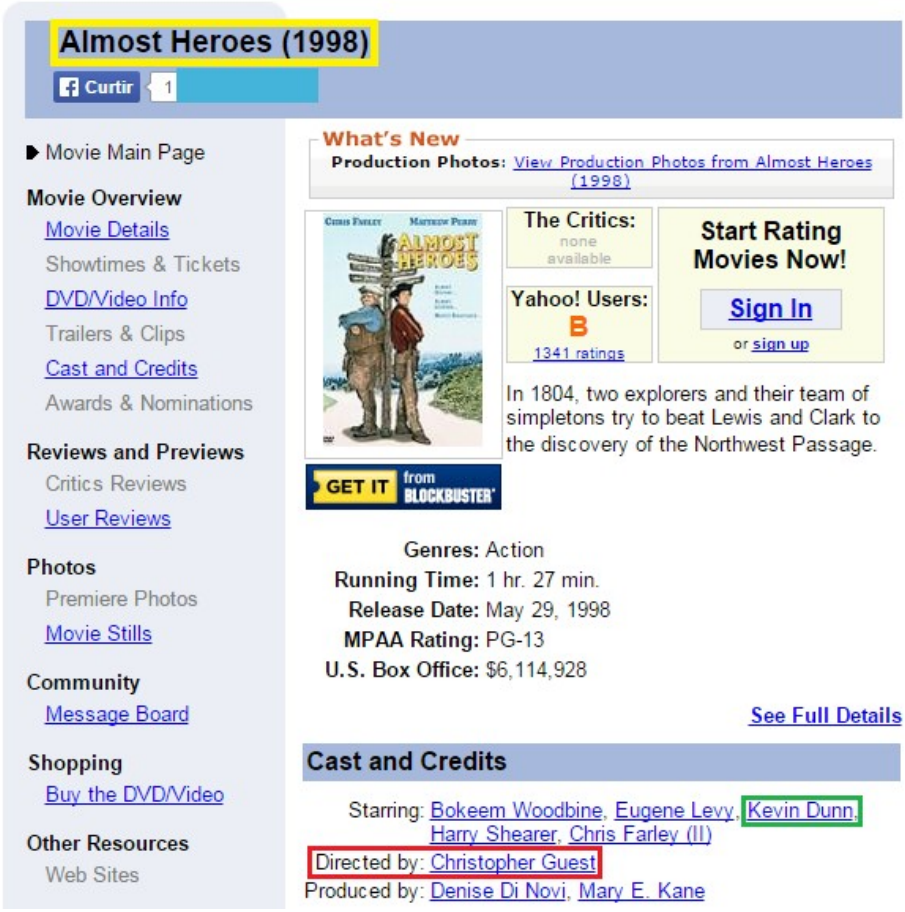
Devido ao limite da quantidade de valores de exemplo dos métodos, esses valores podem não estar em intersecção com os valores das páginas de treinamento, em determinados sites. Em alguns casos, quando há redundância em nível de instância, podem haver valores correspondentes nos exemplos, mas não ocorrer uma correspondência de instância. Por exemplo, supondo a lista de valores de exemplo apresentados na Tabela 4.8 e uma página de treinamento na Figura 4.10. Em uma busca de similaridade, o valor "*Kevin Dunn*" encontrado na página, destacado em verde, seria o mais próximo do valor de exemplo "*Kevin Nolan*", do atributo Diretor, apesar de não corresponder ao atributo Diretor da instância da página. No caso, como não existem outros valores de exemplo, o caminho incorreto do valor "*Kevin Dunn*" seria escolhido como ideal, apesar do valor correto da instância ser "*Christopher Guest*", destacado em vermelho. No caso em que o método use a redundância em nível de instância, e compare a instância do valor de exemplo com a instância da página, o atributo Título, "*Filme X*", do exemplo, não pertenceria à mesma instância da página, de valor "*Almost Heroes (1998)*" (destacado em amarelo). Nesse caso, o treinamento do método não retorna um caminho e não realiza a extração do atributo Título.

Tabela 4.8: Exemplos de instância

Diretor	Gênero	Classificação	Título
Kevin Nolan	Action	PG-13	Filme X

Fonte: Elaborado pelo autor.

Figura 4.10: Página da base de filmes



Fonte: https://www.yahoo.com/movies

A qualidade da extração do FindAttrPos foi prejudicada por esse caso, devido a redundância em nível de instância do método. Em algumas páginas, o método não encontra valores similares aos valores de exemplo de instância e não retorna um caminho para o atributo buscado. A revocação do FindAttrPos foi afetada pois o método não extraiu valores em algumas páginas.

O caso de falha de valores fora da intersecção entre exemplos e site aconteceu em 1 site (Yahoo) da base dados de filmes. O caso de falha não ocorreu na base de dados de projetos de software.

Uma solução para esse caso seria aumentar a quantidade de exemplos inseridos na entrada dos métodos, entretanto isso pode acarretar um aumento considerável de tempo de execução.

5 CONCLUSÃO

O presente trabalho analisou três métodos de extração de dados da Web baseado em redundância de conteúdo, implementando dois deles. Os métodos foram analisados de acordo com a coleta de dados em duas bases de dados, sendo uma delas criada, com o seu gabarito, neste trabalho.

As bases de dados usadas neste trabalho possuem características diferentes, como a possibilidade de atributos omissos ou com múltiplos rótulos, alterando assim a posição desses atributos dentro de um mesmo site. Além disso, a informação de rótulo pode não estar presente em uma base, criando um desafio para os métodos tratarem em sua execução. As bases de dados usadas foram úteis para analisar o comportamento específico de cada método, baseado na notação que cada um deles usa para encontrar uma posição de atributo dentro de uma página. Essa notação, caminho com índice ou sem índice, demonstra a suposição de que os atributos mantenham-se, ou não, em uma posição fixa entre as páginas de um site. Outra característica dos métodos está relacionada a busca por similaridade entre os valores nas páginas dos sites e nos valores de exemplos inserido na entrada dos métodos. Dependendo do caso é necessária uma busca por similaridade mais robusta, que ignora partes de um nodo textual para comparar somente os valores relevantes. A informação de rótulo busca impedir a coleta de valores incorretos, analisando por sua vez, se o valor pertence ao atributo desejado. Já outra característica muito importante é a capacidade de lidar com a redundância em nível de instância, ou seja, de identificar se um valor pertence a uma instância dentro de uma página ou se o valor coletado é um ruído, constantemente presentes nos sites. Essa característica é importante, por exemplo, para aumentar a revocação do método, impedindo-o de coletar valores referentes à instâncias diversas à que a página descreve. Apesar dessas características surgirem para aumentar a qualidade de extração dos métodos, as soluções criadas podem aumentar a quantidade de informação inserida como entrada para os métodos. Por exemplo, a redundância em nível de instância, necessita da informação dos exemplos identificados dentro de cada instância, obrigando o usuário do método a configurar cada instância específica. Em

contraponto, um meio mais simples para o usuário é inserir os valores de exemplo fora do contexto da instância.

Os resultados dos testes mostraram uma melhor qualidade de extração do AERD na base de dados de projetos de software[19]. Isso se deve principalmente ao uso da notação de caminho sem índice do método, mais flexível para lidar com atributos que variam de posição entre as páginas (característica dessa base de dados). O FindAttrPos obteve melhores resultados, em relação ao AERD, na base de filmes. Nessa base, a maioria dos sites não varia a posição dos atributos. A notação de caminho com índice, do FindAttrPos, pressupõe essa estrutura fixa dos atributos. Além disso, a inexistência de rótulo para alguns atributos da base de filmes afetou a extração do AERD, que necessita dessa informação. Entretanto, o WEIR obteve resultados superior aos outros na base de filmes, devido à sua redundância em nível de instância mais robusta, capaz de identificar a instância da página e extrair os valores referentes a essa instância.

A análise se mostrou interessante para evidenciar o comportamento dos métodos, entretanto, no decorrer da análise, ficou clara a possibilidade de realizar mais testes, nas mesmas bases usadas e em outras bases, variando as configurações de parâmetros. Outras bases produzem outros problemas e possivelmente novas soluções. Uma possibilidade, por exemplo, seria aliar as soluções dos métodos em uma nova proposta, tirando proveito das vantagens dos mesmos, dependendo das características da base de dados em que se pretende extrair informação. Dessa forma se abrem caminhos para trabalhos futuros.

[19]Apêndice 2– Recomendação do uso dos métodos

REFERÊNCIAS

AGRAWAL, R.; SRIKANT, R. Fast Algorithms for Mining Association Rules in Large Databases. In: VLDB, 1994, San Francisco, CA, USA. **Anais...** Morgan Kaufmann Publishers Inc., 1994. p.487–499.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval**: the concepts and technology behind search. [S.l.]: Addison Wesley Professional, 2011.

BLANCO, L. et al. Supporting the automatic construction of entity aware search engines. In: **WIDM**, 2008. Anais... ACM, 2008. p.149–156.

BRONZI, M. et al. Extraction and Integration of Partially Overlapping Web Sources. **PVLDB**, [S.l.], v.6, n.10, p.805–816, 2013.

CHANG, C.-H. et al. A Survey of Web Information Extraction Systems. **IEEE Trans. On Knowl. and Data Eng.**, Piscataway, NJ, USA, v.18, n.10, p.1411–1428, Oct. 2006

GIBSON, D.; PUNERA, K.; TOMKINS, A. The volume and evolution of web page templates. **WWW**, 2005.

GRAVANO, L. et al. Text Joins in an RDBMS for Web Data Integration. In: WWW, 2003, New York, NY, USA. **Anais...** ACM, 2003. p.90–101.

GULHANE, P. et al. Exploiting Content Redundancy for Web Information Extraction. **PVLDB**, [S.l.], v.3, n.1, p.578–587, 2010.

HAO, Q. et al. From One Tree to a Forest: a unified solution for structured web data extraction. In: SIGIR, 2011, New York, NY, USA. **Anais...** ACM, 2011. p.775–784.

HEUSER, C. A. **Projeto de Banco de Dados**. 6. ed. Porto Alegre: Bookman, 2009.

LI, X. et al. Exploiting Attribute Redundancy in Extracting Open Source Forge Websites. In: CYBERC, 2012, Washington, DC, USA. **Anais...** IEEE Computer Society, 2012. p.13–20.

MANICA, E. **Um processo para descoberta e extração de dados de páginas-instância**. 2015. 95 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2015.

MICHEL, M.; ALLEGRETTI, R.; MANICA, E.; GALANTE, R. **Implementação e Experimentos com o método Content Redundancy**. Artigo apresentado na disciplina INF01069 – TÓPICOS ESPECIAIS EM COMPUTAÇÃO XXIX – PESQUISA EM SISTEMAS DE INFORMAÇÃO. Dezembro de 2013.

SIEGEL, S.; CASTELLAN, N. **Nonparametric Statistics for the Behavioral Sciences**. [S.l.]: McGraw-Hill, 1988. (McGraw-Hill international editions. Statistics series).

APÊNDICE 1– VALORES DO TESTE P DE WILCOXON PARA AS MÉTRICAS

Os valores P são referentes aos valores individuais obtidos das métricas dos métodos AERD e FindAttrPos, implementados pelo autor.

Tabela: Valores do teste P de Wilcoxon para as métricas

	Projetos de software	Filmes
Revocação	0,000027	0,097835
Precisão	0,000456	0,007614
F1	0,000056	0,002112

Fonte: Elaborado pelo autor.

APÊNDICE 2– RECOMENDAÇÃO DO USO DOS MÉTODOS

Os métodos recebem recomendações para uso de acordo com as características das bases de dados testadas neste trabalho. Os métodos AERD e FindAttrPos foram testados nas duas bases de dados (projetos de software e filmes). O WEIR foi testado somente para a base de filmes, dessa forma, somente é recomendado o uso nessa base, já que o método obteve resultados superiores em relação aos outros métodos, na avaliação dos experimentos.

Tabela: Recomendação do uso dos métodos nas bases de dados

	Características	Base de dados
AERD	Existência de rótulos para os atributos; Pouco ruído.	Projetos de software
FindAttrPos	Não variação da posição dos atributos em um site; Atributos sem múltiplos rótulos.	Filmes
WEIR	Existência de rótulos preferível; Variação ou não da posição dos atributos em um site;	Filmes

Fonte: Elaborado pelo autor.