Coleta e visualização de dados disponíveis em repositórios públicos referentes a produção científica de pesquisadores

Arthur M. Branco[1], **Carina F. Dorneles**[1]

[1]Departamento de Informática e Estatística - INE
Universidade Federal de Santa Catarina - UFSC/Florianópolis

arthur.m.branco10@hotmail.com, carina.dorneles@ufsc.br

Abstract. *Data that reflect the scientific production of researchers have an inestimable value for several applications. Several repositories index articles and make them available for consultation, such as DBLP, Research Gate, and Google Scholar. Although the data are available in several public repositories, this data's collection and local persistence can benefit specific applications. This article presents a proposal for a data collector from three public repositories: DBLP, Research Gate, and Google Scholar, and their subsequent persistence in a relational database. In addition, a visualization interface for the collected data is also presented.*

Resumo. *Dados que refletem a produção científica de pesquisadores têm valor inestimado para diversas aplicações. Há diversos repositórios que indexam os artigos e os disponibilizam para consultas, tais como DBLP, Research Gate e Google Scholar. Apesar dos dados estarem disponíveis nos diversos repositórios públicos, a coleta e persistência local desses dados pode ser de grande utilidade para certas aplicações.Este artigo apresenta uma proposta de coletor de dados de três repositórios públicos: DBLP, Research Gate e Google Scholar, e sua posterior persistência em um banco de dados relacional. Além disso, é apresentada também uma interface de visualização para os dados coletados.*

1. Introdução

Dados que refletem a produção científica de pesquisadores estão disponíveis em muitos repositórios na Web, como ACM Portal[1], IEEE[2], Elsevier[3], entre outros, pois as obras publicadas estão sempre associadas aos editores. Além disso, há diversos repositórios que indexam os artigos e os disponibilizam para consultas, tais como DBLP[4], Research Gate[5] e Google Scholar[6].

Estes dados têm alta relevância para diversas aplicações, tais como análise de coautoria [Brandão and Moro 2017, Kumar 2015, Lopes et al. 2010], *Expertise Retrieval* [Balog et al. 2012], desambiguação de nome de autor [Ferreira et al. 2012, Pereira et al. 2009], entre outras [Färber 2020, Malisart 2009]. Apesar dos dados estarem

[1]acm.org
[2]IEEE.org
[3]elsevier.com
[4]https://dblp.org
[5]https://www.researchgate.net/
[6]https://scholar.google.com.br/

disponíveis nos diversos repositórios públicos, a coleta e persistência local desses dados tem grande utilidade para realização de experimentos de novas técnicas das aplicações supracitadas.

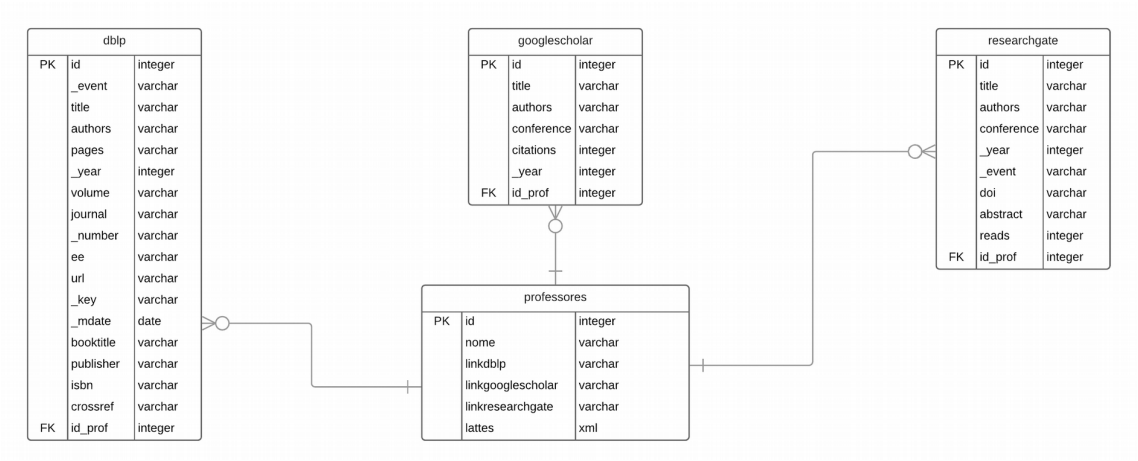
Este artigo apresenta uma proposta de coletor de dados de três repositórios públicos: DBLP, Research Gate e Google Scholar, e sua posterior persistência em um banco de dados relacional. Como cada um dos repositórios apresenta esquemas de dados diferentes, foi necessário realizar uma adequação dos dados antes de serem inseridos no banco de dados. Além do coletor propriamente dito, é apresentada um interface de visualização, que permite que os dados coletados sejam visualizados em diferentes abas de apresentação. O coletor pode ser rodado sobre os repositórios de dados a qualquer momento para extrair novas instâncias, que podem ser persistidas em banco de dados relacional e posteriormente consultadas e usadas em experimentos. O protótipo também lê o documento XML do currículo Lattes do pesquisador, a fim de permitir uma análise de quão atualizado se encontram os currículos quando comparados com repositórios públicos. Esta funcionalidade foi desenvolvida após constatação de que alguns pesquisadores não atualizam seus Lattes como deveria, prejudicando muitas vezes, os programas de pós-gradução nos quais atuam.

O artigo está organizado como segue. Na Seção 2, é apresentado o processo de funcionamento do *crawler* e do extrator propostos. Na Seção 2, é descrita a base de dados e suas principais características, bem como a interface de acesso. Na Seção 4 são apresentadas as conclusões e as direções futuras.

2. Extraindo os dados das fontes

Esta seção apresenta a lógica de extração de dados de cada uma das fontes consideradas. A Figura 1 apresenta o esquema do banco de dados e consequentemente os atributos extraídos[7] em cada uma das fontes de dados.

Figura 1. Esquema do Banco de Dados



2.1. DBLP

A fonte do DBLP pode ser considerada a mais bem estruturada, e consequentemente, a menos desafiadora das três fontes de dados consideradas. Isso porque os dados são dispo-

[7] https://github.com/arthurmbranco/CrawlerTCC/tree/master/files/data

nibilizados através de um documento XML, que facilita a extração e posterior persistência em um banco de dados relacional. Portanto, a ideia básica é fazer uma conversão simples do documento XML para o esquema do banco de dados, considerando os atributos definidos para a DBLP citados na Figura 1.

De forma geral, existem dois tipos de dados, dados pessoais do pesquisador e dados sobre as obras, e que foram utilizados para montar a base dados. O elemento `person` armazena dados pessoais do pesquisador, e as obras são armazenadas dentro de elementos `r` e armazenam dados sobre as publicações dos pesquisadores, apresentando os seguintes elementos possíveis: `article`, `inproceedings`, `proceedings`, `book`, `incollection`, `phdthesis`, `mastersthesis` e `www`.

2.2. Google Scholar

Os dados do Google Scholar estão em formato HTML, portanto é necessário navegar pelos elementos específicos que guardam informações de cada obra. As obras estão contidas dentro do elemento `gsc_a_tr`, que possui três elementos filhos: `gsc_a_t` que possui metadados separados sobre o título da obra, autores, evento, conferência e universidade dependendo de tipo da natureza da obra; `gsc_a_c` que contém número de citações da obra; e `gsc_a_y` que possui o ano da obra.

Foi necessária a criação de um algoritmo para paginação visto que cada página apresenta no máximo 100 resultados de obras, e um usuário que usa o *browser* precisa clicar em um botão com a *label* "MOSTRAR MAIS" e através da execução de código *JavaScript* são carregado novos dados na página, porém o *crawler* só consegue ler páginas HTML e não consegue executar o código *JavaScript*. Como forma de contornar este problema foi feita uma análise nas requisições feitas pela rede através do console e foi descoberto que existe um padrão para a construção da URL da página que possibilita que o *crawler* acesse todas as obras de um pesquisador mesmo que possua mais de 100 obras.

Um problema que foi encontrado na estrutura dos dados deste repositório é que existe um tamanho limite na string que representa os autores e o nome e dados da conferência ou evento e isso faz com que os dados sejam cortados caso o limite seja extrapolado. Para resolver este problema é necessário extrair um link do atributo `datahref` que está armazenado no elemento `gsc_a_t`, concatenando a string do GoogleScholar com o link extraído é possível acessar uma nova página que pode ser acessada pelo usuário ao clicar no título da obra, esta nova página contém dados sobre: autores, data de publicação, conferência, páginas, editora, resumo e total de citações. Os dados apresentados nesta página estão completos, sem cortes, e também pode se encontrar dados que não estão presentes na outra página que irão ser persistidos no banco de dados.

2.3. Research Gate

O repositório do ResearchGate, semelhante ao GoogleScholar, também contém seus dados estruturados em HTML e é muito mais rico em quantidade de dados que oferece sobre as obras. As obras são acessadas através do elemento que possui label: `nova-v-publication-item_stack nova-vpublication-item_stack--gutter-m`.

A estrutura dos filhos do nodo citado acima pode variar, algumas obras possuem um conjunto de imagens referentes ao trabalho que foi feito e outras obras

não possuem imagens. Como todos os filhos possuem o elemento com a label `nova-vpublication-item_stack-item` o acesso a eles é feito pelo seu índice e por isso é preciso verificar se existe uma imagem associada a obra para coletar os metadados sem o código apresentar exceções. A partir do elemento inicial é possível extrair informação do título, autores, ano da obra, URL para uma página que contém mais detalhes da obra e o tipo de obra.

O acesso à URL contendo detalhes sobre a obra que está sendo analisada é para que o crawler salve a maior quantidade possível de dados relevantes sobre a obra. A página contém metadados que não estavam disponíveis na página principal do pesquisador, tais como: dados sobre a conferência, evento ou periódico onde foi publicada a obra; número do DOI; abstract da obra; e número de leituras desta obra através do site do ResearchGate.

Também foi necessário implementar um mecanismo de paginação visto que o ResearchGate só carrega 100 obras por página. Para navegar entre as páginas basta concatenar a string `/n` sendo n o número da página que se deseja acessar com a URL da página inicial do perfil do pesquisador. Por exemplo, se o crawler estiver acessando os dados de um pesquisador cuja URL seja `https://www.researchgate.net/profile/Professor` e for necessário acessar a segunda página do seu perfil a URL resultante da concatenação seria: `https://www.researchgate.net/profile/Professor/2`.

A página possui um componente que indica a quantidade de "Research Items" contidos no perfil de cada pesquisador que pode ser acessado pelo elemento `nova-o-stack_item pagination--top` para extrair o número de obras. O algoritmo necessita de uma variável de controle "n" que tem seu valor inicial igual a 1. Uma variável "itensPagina" é inicializada em 0 e enquanto o valor de "itensPagina" for menor que o número de "Research Items" o valor de itensPagina é incrementado em 100 e o valor de "n" é incrementado em 1 e o acesso à nova página é realizado. Quando o valor de "itensPagina" superar o valor de "Research Items" significa que não existe mais nenhuma página para iterar e o algoritmo encerra a execução.

2.4. Implementação

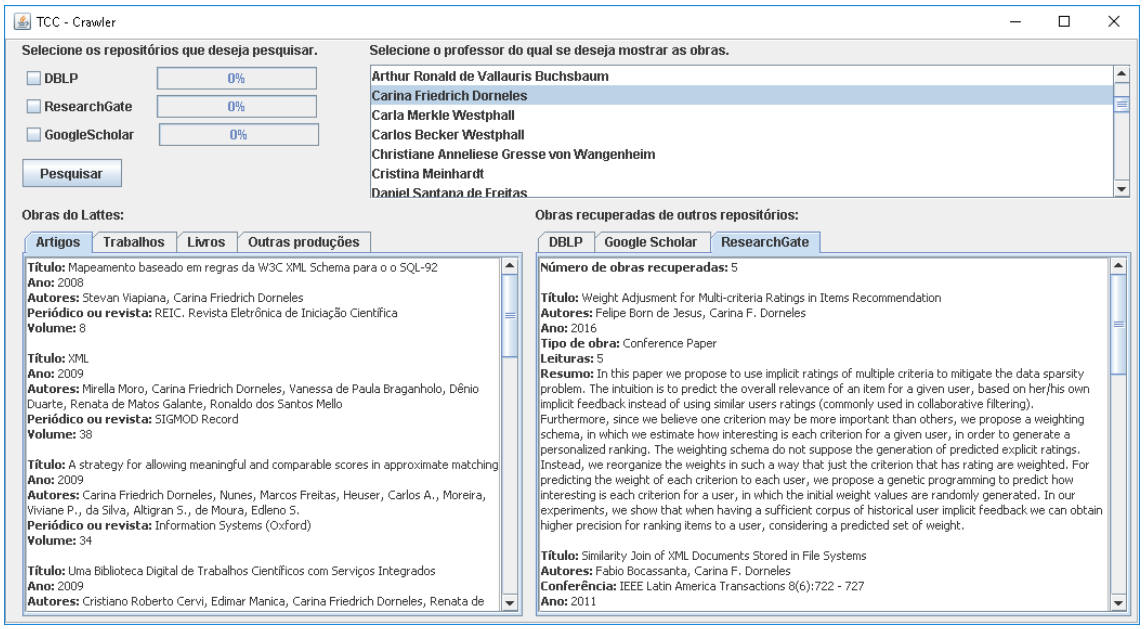
O coletor de dados[8] foi desenvolvido utilizando a linguagem de programação JAVA com o auxílio da IDE (Integrated Development Environment) Eclipse e foi usado o banco de dados PostgreSQL juntamente a ferramenta pgAdmin 4 que facilita a visualização dos dados do banco através de uma interface acessível por um browser. Foram usadas também três bibliotecas externas para o desenvolvimento do projeto: (i) Jsoup, que é uma biblioteca em JAVA desenvolvida para trabalhar com páginas HTML e possui uma estrutura para manipulação e extração de dados; (ii) Java-string-similarity, que é uma biblioteca que implementa diversas métricas de similaridade e distância de strings; e (iii) PostgreSQL JDBC Driver9 é a biblioteca usada para permitir que o programa em JAVA se conecte com o banco de dados PostgreSQL.

[8]https://github.com/arthurmbranco/CrawlerTCC

3. Interface de acesso

A Figura 2 apresenta a interface de acesso aos dados coletados. Através do componente *checkbox* é possível selecionar os repositórios em que se deseja fazer a pesquisa. Ao clicar no botão "Pesquisar" a busca por novas obras é iniciada. No canto superior direito da interface é possível selecionar todos os professores contidos no banco de dados. Caso um professor seja selecionado, inicia-se uma busca por todas as obras recuperadas e também todas as obras do Lattes[9] do pesquisador contidas nas seções de: artigos, trabalhos, livros e outras produções. Após isso as duas áreas de texto presentes na parte inferior da interface são populadas com as obras.

Figura 2. Interface de acesso aos dados coletados



4. Conclusões e Trabalhos futuros

O artigo apresentou um coletor de dados cujo intuito é unificar os dados das obras contidas em diferentes repositórios em uma única aplicação. Foi recuperada uma quantidade considerável de novas obras advindas dos repositórios da DBLP, GoogleScholar e ResearchGate. A ferramenta também apresenta uma interface simples, sendo possível com apenas alguns cliques fazer o crawler executar a busca por novas informações e em seguida mostrar todos os dados já existentes e os recuperados de forma unificada.

Alguns direcionamentos de trabalhos futuros e ideias que visam melhorar a qualidade da aplicação: (i) refinamento na seleção dos dados extraídos: o crawler itera sobre todas as obras pertencentes aos professores escolhidos nos repositórios selecionados, porém alguns dados encontrados, por exemplo, no site do GoogleScholar, podem ser considerados como pouco ou nada úteis visto que possuem um baixo valor semântico, assim é possível criar algum método que filtre os dados antes de compará-los; (ii) otimização

[9]O protótipo foi desenvolvido, inicialmente, com o objetivo de verificar o quanto os pesquisadores atualizam seus currículos Lattes, e para isso, é feito um *matching entre os dados do Lattes com os dados coletados*

do tempo de acesso aos repositórios: Para evitar problemas com bloqueio de IP, o crawler rotaciona entre vários proxies acessando as páginas com IP´s diferentes, porém nos experimentos notou-se que muitos proxies são instáveis e possuem uma conexão lenta, fator que contribui de forma muito negativa para o tempo de pesquisa das obras. Se esta parte for desenvolvida de maneira adequada através, por exemplo, do uso do TorBrowser ou de outras maneiras não pensadas, pode ser possível reduzir o tempo de busca em dezenas ou até centenas de vezes; (iii) automatização de tarefas: Infelizmente é necessário que o crawler seja configurado por um humano antes de funcionar corretamente, devido a dificuldade de resolver os captchas no site da Plataforma Lattes é necessário que os currículos sejam baixados manualmente, é necessário também popular o banco de dados com o nome dos professores e os links referentes as suas obras em cada um dos repositórios. Se forem automatizadas essas duas tarefas então o crawler se torna totalmente independente de interferência humana para funcionar; e (iV) expansão do escopo de busca: Acrescentar novos repositórios irá aumentar o número de obras recuperadas e contribuir com a atualização dos currículos investigados.

Referências

Balog, K., Fang, Y., de Rijke, M., Serdyukov, P., and Si, L. (2012). Expertise retrieval. *Foundations and Trends® in Information Retrieval*, 6(2–3):127–256.

Brandão, M. A. and Moro, M. M. (2017). The strength of co-authorship ties through different topological properties. *Journal of the Brazilian Computer Society*, 23(1):5.

Färber, M. (2020). Analyzing the github repositories of research papers. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, page 491–492, New York, NY, USA. Association for Computing Machinery.

Ferreira, A. A., Gonçalves, M. A., and Laender, A. H. (2012). A brief survey of automatic methods for author name disambiguation. *Acm Sigmod Record*, 41(2):15–26.

Kumar, S. (2015). Co-authorship networks: A review of the literature. *Aslib Journal of Information Management*, 67:55–73.

Lopes, G., Moro, M., Wives, L., and Palazzo Moreira de Oliveira, J. (2010). Cooperative authorship social network. volume 619.

Malisart, A. (2009). Researcher profile: A web 2.0 application for visualising research communities. In *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops*, IWPSE-Evol '09, page 145–152, New York, NY, USA. Association for Computing Machinery.

Pereira, D. A., Ribeiro-Neto, B., Ziviani, N., Laender, A. H., Gonçalves, M. A., and Ferreira, A. A. (2009). Using web information for author name disambiguation. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 49–58.