UNIVERSIDADE FEDERAL DE SANTA CATARINA

FERRAMENTA PARA COLETA E COMPARAÇÃO DE DADOS DE PUBLICAÇÕES ACADÊMICAS DOS PROFESSORES COM O CURRÍCULO LATTES

ARTHUR MACHADO BRANCO

Florianópolis - SC

2018/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

FERRAMENTA PARA COLETA E COMPARAÇÃO DE DADOS DE PUBLICAÇÕES ACADÊMICAS DOS PROFESSORES COM O CURRÍCULO LATTES

ARTHUR MACHADO BRANCO

Trabalho de conclusão de curso apresentado
como parte dos requisitos para obtenção do grau
em Bacharel em Ciências da Computação.

Florianópolis - SC
2018/2

ARTHUR MACHADO BRANCO

FERRAMENTA PARA COLETA E COMPARAÇÃO DE DADOS DE PUBLICAÇÕES ACADÊMICAS DOS PROFESSORES COM O CURRÍCULO LATTES

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau em Bacharel em Ciências da Computação.

Orientador(a): Rodrigo Gonçalves

Banca examinadora

Carina Friedrich Dorneles

Ronaldo dos Santos Mello

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE REDUÇÕES

CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
INE	Departamento de Informática e Estatística
XML	*Extensible Markup Language*
SQL	*Structured Query Language*
HTTP	*HyperText Transfer Protocol*
URL	*Uniform Resource Locator*
CSV	*Comma-separated Values*
PDF	*Portable Document Format*
DOM	*Document Object Model*
GUI	*Graphical User Interface*
IP	*Internet Protocol*
CV	*Curriculum Vitae*

RESUMO

Atualmente uma forma de avaliar as competências e a quantidade de obras produzidas por pesquisadores é através do currículo lattes que foi criado pelo CNPq e serve como um padrão nacional para o registro da vida acadêmica e portanto é adotado por um grande número de instituições no país devido ao rico grau de detalhes e informações que estão contidos em si [1].

Tendo em vista a importância que o currículo lattes representa para avaliar as competências profissionais, é de interesse dos pesquisadores manter os seus currículos atualizados e por isso este trabalho tem o objetivo de recuperar informações faltantes no currículo lattes a fim de torná-lo mais completo .

Com o uso de um *crawler* é possível extrair informações relevantes de outras fontes de dados a fim de complementar o currículo lattes, comparando as obras contidas em três repositórios (DBLP, GoogleScholar e ResearchGate) com as obras contidas no lattes é possível criar uma base de dados que contenha novos dados que não estão contidos no lattes.

Com a criação de uma aplicação que integre e unifique os dados do lattes com as novas obras que forem encontradas o usuário pode facilmente visualizar as novas obras e atualizar o seu currículo caso seja necessário.

Palavras-chave: *Crawler*, Currículo lattes, Recuperação de dados

1. INTRODUÇÃO

Existem diversas métricas para avaliar um pesquisador dentro do mundo acadêmico como: número total de publicações, número de citações, *H-index, G-index*, *HC-index* e *HI-norm* [2]. Para que se aplique as métricas descritas de forma ideal, é necessário conter o conjunto completo de obras produzidas por um pesquisador.

Atualmente o currículo lattes é um padrão nacional usado por grande parte dos pesquisadores para armazenar dados referentes a todo seu histórico de vida acadêmica e profissional [1]. Estes dados são adicionados de forma manual ao currículo lattes e podem não refletir todo o potencial do pesquisador pois não é comum a eles manterem o currículo totalmente atualizado constantemente.

O currículo lattes é de grande importância e através dele é possível que os profissionais possam garantir sua participação em grupos de pesquisa, participar de eventos e conferências e conseguir vagas para trabalhar em alguma determinada área. Assim surge a necessidade de manter um currículo atualizado que é usada como motivação para a criação da ferramenta desenvolvida nesta obra.

A internet disponibiliza a todos o acesso a quantidades gigantes de dados de vários tipos, incluindo vários sites especializados em armazenar dados sobre as produções dos pesquisadores. Para trabalhar com tantos dados de forma eficaz precisamos usar ferramentas de software especializadas em coletar dados da web, que são conhecidas como *webcrawlers* [3].

Apesar de o auxílio de um *webcrawler* ser essencial neste tipo de tarefa, seu uso tem que ser bem planejado, se a pesquisa pelos dados for feita de forma mais abrangente pesquisando através de motores de busca que indexam quantidades gigantes de dados a qualidade destas buscas será pior. Quando o domínio é muito extenso as buscas por dados tem que ser feitas de maneira mais genérica possível, por isso este trabalho se propõe a criar um *crawler* focado em alguns sites específicos que contém apenas os dados de interesse, que são as obras dos pesquisadores.

A coleta de dados será feita a partir de três sites com destaque no mundo acadêmico que são: DBLP, GoogleScholar e ResearchGate. É possível comparar seus dados com os dados de publicações do currículo lattes a fim de criar uma

visão unificada que tem como objetivo armazenar a maior quantidade possível de produções dos pesquisadores para que suas competências possam ser avaliadas por completo.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo detalhar os conceitos e definições teóricas utilizados para o desenvolvimento deste trabalho. Será detalhado o conceito de um *webcrawler,* detalhamento do currículo lattes e dos sites usados como nova fonte de dados que serão comparadas e unificadas ao mesmo, estrutura e extração de dados na web. Deste capítulo em diante a referência a um *webcrawler* será feita na sua forma reduzida comumente conhecida como *crawler*.

2.1 *CRAWLER*

Os *crawlers* são ferramentas em forma de software que vasculham a web em busca de informações que contribuam para determinado domínio. Eles são muito usados por motores de busca para visitar páginas e salvá-las a fim de realizar um pré-processamento e indexá-las em um banco de dados promovendo uma pesquisa mais rápida pelos usuários do motor de busca [4].

Com o aumento da demanda de automatização de tarefas os *crawlers* se tornaram cada vez mais populares e com isso muitos sites contém um arquivo chamado “robots.txt” que define algumas regras como: quais diretórios e páginas podem ou não ser indexadas e uma *crawl rate* que seria a velocidade com o que o *crawler* pode indexar as páginas do site. Esta necessidade é advinda do fato de que os *crawlers* consomem muito recurso do servidor das páginas indexadas, então é necessário um controle sobre eles para que não haja sobrecarga no servidor e para que não causem lentidão para os usuários humanos que também acessam as páginas.

2.1.1 *CRAWLER* DE PROPÓSITO GERAL

Os *crawlers* genéricos que são criados para visitarem vários tipos páginas necessitam de um cuidado maior ao serem escritos por abrangerem um maior

domínio com páginas que podem ser totalmente diferentes entre si. Os *crawlers* necessitam de três informações básicas para funcionarem.

1. Uma lista de URLs não visitadas populadas inicialmente com algumas páginas que servirão de sementes para pesquisar novas páginas e dados.
2. Uma lista de URLs já visitadas para impedir que o *crawler* entre em loop e possa funcionar de maneira adequada.
3. Uma lista de regras para indexar novas páginas a serem visitadas, assim evitando que conteúdos que não sejam de interesse sejam indexados.

Com estas três informações é possível aplicar um algoritmo genérico que será ilustrado na figura 1.

```
WHILE(!listaDeURL.empty())
    Selecionar uma URL da lista de páginas não visitadas;
    Remover URL da lista de páginas não visitadas
    e adicionar a lista de páginas visitadas;
    Buscar conteúdo desejado;
    Persistir em um banco de dados o conteúdo de interesse;

    IF(conteúdo == HTML)
        Extrair URL dos links;
        FOR EACH (URL)
            Se estiverem de acordo com as regras definidas e a URL
            não estiver contida na lista de visitadas ou de não visitadas
            então adicione a lista de URLs não visitadas;
        END FOR
    END IF
END WHILE

```

Figura 1 - Pseudo-código ilustrando um modelo de algoritmo para um *crawler genérico*

Visto que a quantidade de dados em páginas disponíveis na web é imensa, um bom *crawler* precisa de uma lista de regras bem definida para indexar o maior número possível de páginas interessantes, caso contrário ele pode passar mais tempo procurando por páginas que não acrescentam em nada ao trabalho. Quanto mais complexo for o *crawler* maior tem que ser a preocupação com problemas como: falta de memória por indexar uma quantidade gigante de páginas, falta de disco para persistir todos os dados, lidar com códigos HTML mal formatados e páginas que contenham erros HTTP.

2.1.2 CRAWLER FOCADO

Este tipo de *crawler*, como o nome sugere, é focado em páginas específicas, sendo assim a precisão dos dados buscados aumenta consideravelmente visto que com um domínio menor é possível estudar as páginas e organizar quais dados podem ou não ser coletados [5]. Este trabalho se encaixa nesta categoria e visa coletar dados de três fontes diferentes que serão abordadas nas próximas seções e usar estes dados para fazer comparações com o currículo lattes dos professores e incrementá-lo.

A complexidade neste tipo de *crawler* é muito menor já que a lista de regras vai ser especializada para um certo número de páginas conhecidas e as URL´s que são vasculhadas normalmente são subpáginas das URL´s semente contidas na lista de URL iniciais. Adicionalmente é mais fácil testar e garantir que o *crawler* não contenha erros caso o domínio seja todo conhecido previamente, que é o caso deste trabalho.

2.2 PLATAFORMA LATTES

A plataforma Lattes[1] foi criada por interesse do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) em integrar bases de dados de currículos, grupos de pesquisa e instituições em um único sistema de informações, ela tem grande importância no mundo acadêmico e, sendo um padrão nacional para o armazenamento dos chamados “currículos lattes” que registram vários tipos de dados sobre as produções e participações em eventos acadêmicos dos pesquisadores e estudantes do Brasil.

Hoje em dia é do interesse de vários pesquisadores manter seu currículo lattes atualizado tornando ele uma fonte rica em informações. Todos esses currículos são de fácil acesso para qualquer pessoa que tenha interesse e eles podem também ser salvos em XML, que é um formato conveniente e estruturado.

Os currículos lattes surgiram da necessidade de se ter um currículo padrão e rico em detalhamento de informações que registram o histórico de produções dos

1http://lattes.cnpq.br/

pesquisadores e alunos. Por meio do currículo lattes é possível mensurar as capacidades e o nível de envolvimento deles em relação aos tópicos estudados.

2.2.1 ESTRUTURA E UM CURRÍCULO LATTES

Através da plataforma lattes é possível visualizar todos os dados referentes aos currículos lattes que são usados como a base de comparação de dados neste trabalho. Será detalhado neste tópico a sua estrutura com uma breve descrição dos campos.

1. **Identificação:** Possui nome completo e nomes usados em citações bibliográficas.
2. **Endereço:** Possui dados do endereço profissional, formas de contato e URL da *homepage*.
3. **Formação acadêmica/titulação:** Possui dados sobre a formação acadêmica como: tempo de conclusão, instituição, título da obra, orientador, palavras-chave, área e subárea do trabalho.
4. **Atuação Profissional:** Possui dados das instituições em que ocorreu a atuação profissional, duração do vínculo e lista as atividades feitas no período.
5. **Linhas de Pesquisa:** Lista todas as linhas de pesquisa de interesse do profissional.
6. **Projetos de Pesquisa/Desenvolvimento:** Lista todos os projetos de pesquisa e desenvolvimento concluídos ou em execução informando o título, descrição, integrantes do projeto, financiadores e número de orientações geradas a partir do projeto.
7. **Áreas de atuação:** Lista as áreas de atuação do profissional, indicando a grande área, área, subárea e especialidade.
8. **Idiomas:** Descreve quais idiomas o profissional tem fluência e qual o nível de fluência.
9. **Prêmios e Títulos:** Lista as premiações e títulos informando a data e evento.
10. **Produções:** Lista de vários tipos de produções como: artigos completos publicados em periódicos, artigos aceitos para publicação, livros e capítulos,

textos em jornais ou revistas, trabalhos publicados em anais de congressos, apresentações de trabalhos, partituras musicais, tradução, prefácio e posfácio e outros tipos de produções bibliográficas.

11. **Bancas:** Lista de participações em bancas de trabalhos de conclusão de curso e comissões julgadores.
12. **Orientações:** Contém dados de todas as orientações de graduação, mestrado e doutorado em andamento e que foram concluídas, possui o título da obra, ano, palavras-chave, autor e instituição.
13. **Eventos:** Lista a participação e organização em eventos, congressos, exposições e feiras.

Para acessar o currículo lattes de cada profissional é necessário validar um CAPTCHA, que funciona como um teste cognitivo em que é necessário preencher um campo com letras que estão inseridas em uma imagem. Por ser uma tarefa difícil de automatizar, neste trabalho é feita a análise sintática de dados em um arquivo XML que foram acessados e baixados manualmente nas páginas dos currículos lattes dos professores .

O XML é gerado obedecendo o padrão chamado DOM, que pode ser visto na figura 2. É representado por uma estrutura em formato de árvore em que os nós guardam objetos e desta forma é conveniente de ser lido[2]. Por ser muito popular possui bibliotecas implementadas para sua leitura escritas em várias linguagens de programação como Java, Python, PHP e etc.

2https://www.w3.org/TR/xml11/

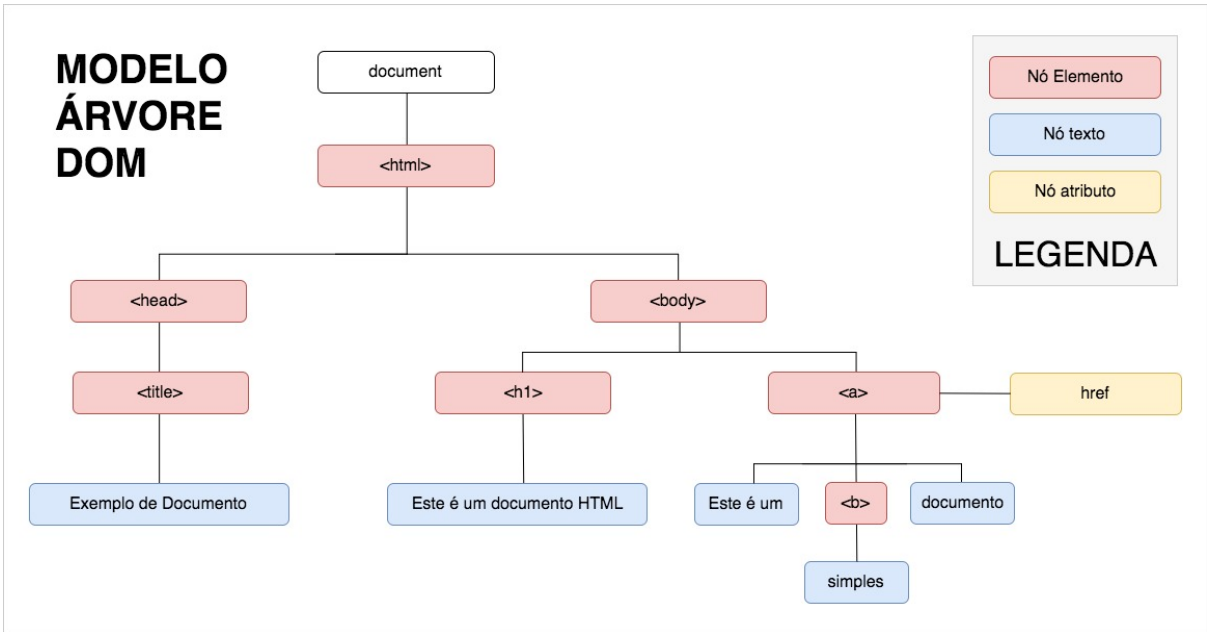


Figura 2 - Exemplo de um modelo de árvore DOM

A disposição dos currículos lattes em XML reflete a grande importância e interesse da comunidade nos dados dos profissionais. Diversas aplicações podem se beneficiar do acesso aos dados, com interesse de integrá-los aos seus sistemas de informação e gerar indicadores, estatísticas sobre as produções dos pesquisadores e usá-los como fonte de comparação com dados advindos de outras fontes, esta última finalidade será o foco principal desta obra.

2.3 REPOSITÓRIOS QUE CONTÊM DADOS SOBRE OBRAS DOS PESQUISADORES

O lattes apesar, de representar de maneira muito completa o currículo dos profissionais e ser um requisito de alta prioridade no mundo acadêmico para demonstrar as competências do profissional é incrementado pelos próprios profissionais. Mesmo sendo do interesse deles armazenar dados sobre todo seu histórico de vida acadêmica, muitas vezes podem se esquecer de atualizar o currículo, principalmente aqueles que são mais ativos e produzem uma quantidade maior de obras.

Outros *sites* também existem com a intenção de armazenarem dados de produções podendo assim conter dados que não estejam presentes no lattes,

ajudando no enriquecimento do currículo. Serão usados no trabalho três sites com este intuito e abaixo será descrito um pouco de cada um deles.

2.3.1 DBLP

A DBLP[3] é uma biblioteca digital especializada em publicações na área da Ciências da Computação, em maio de 2016 contava com mais de 3.3 milhões de publicações indexadas publicadas por mais de 1.7 milhões de autores, ou seja, possui uma base grande de dados para coleta de novas informações.

As publicações aqui não são indexadas pelos próprios pesquisadores mas sim obtidas diretamente das editoras e organizadores de eventos e por isso os dados apresentam alta confiabilidade.

2.3.2 RESEARCHGATE

O ResearchGate[4] é uma rede social feita para cientistas e pesquisadores, usada para compartilhar artigos, publicações e dados de experimentos, fazer e responder perguntas e formar uma rede de contatos. Está entre as maiores redes sociais acadêmicas do mundo e os usuários necessitam de um *e-mail* vinculado a uma instituição reconhecida para se registrarem e poder compartilhar seus dados de pesquisas, artigos, experimentos, código de *software*, apresentações e dados gerais.

Este repositório apresenta uma grande variedade de informações que são manualmente compartilhadas pelos usuários da rede social e podem ser facilmente visualizadas até mesmo por membros que não estejam registrados na rede social.

2.3.3 GOOGLESCHOLAR

O GoogleScholar[5] é um motor de busca que indexa metadados e publicações acadêmicas das mais variadas áreas. Os dados são indexados por um *crawler* que

3https://dblp.uni-trier.de/
4https://www.researchgate.net/
5https://scholar.google.com.br/

vasculha toda a rede em busca de dados relevantes para o mundo acadêmico. As obras são buscadas por ordem de relevância e este motor é bem difundido na comunidade acadêmica por indexar um número imenso de documentos.

É possível também que os pesquisadores criem seu perfil neste site e então o *crawler* irá indexar metadados relacionados ao nome do pesquisador no seu perfil, porém pelo fato de os dados serem indexados de forma automatizada, por um *crawler*, eles apresentam uma confiabilidade muito menor em relação aos dois repositórios citados acima.

2.4 ESTRUTURA DOS DADOS NA WEB

Os dados disponíveis nas páginas web estão disponíveis de forma geral de três maneiras diferentes: estruturados, semiestruturados e não estruturados, esta obra se preocupa em extrair dados semiestruturados das páginas web.

Dados semiestruturados são dados que estão armazenados de uma forma organizada, porém diferente da maneira formal como estão estruturados os dados em bancos de dados relacionais e outros tipos de tabelas de dados. Eles contém um rótulo que serve para identificar a informação e facilitar a compreensão sobre o dado, um bom exemplo são os bancos de dados semiestruturados, que armazenam seus dados no formato de XML ao invés de tabelas relacionais que atribuem uma *label* e um tipo ao dado em cada coluna [6].

As páginas web são estruturadas em HTML, possuem *tags* que ajudam a acessar e rotular os dados, por exemplo a *tag* <table> que estrutura os dados de uma página em uma tabela contendo linhas e colunas ou a tag que serve como um container genérico que é usado para agrupar elementos de categorias diferentes.

2.5 EXTRAÇÃO DE DADOS DA WEB

As páginas web apresentam um grande volume de dados que estão estruturados no formato HTML, porém ele foi criado para apresentar a informação no *browser* de um usuário humano e não para ser lido e interpretado de forma automatizada por sistemas de extração de dados [7].

Algumas páginas possuem um código HTML criado por humanos, o que faz com que elas possuam um design não padronizado e erros estruturais que dificultam a extração de dados, páginas dos diferentes desenvolvedores possuem um design completamente diferente, o que aumenta ainda mais a complexidade de extração.

A extração de dados pode ser feita através dos *wrappers,* que são programas que extraem dados de uma página e organiza-os em um formato relacional adequado. Os *wrappers* possuem um conjunto de regras que decidem o que será extraído em cada página. A maneira mais simples de criar um *wrapper* é através da análise humana por um programador que irá determinar o conjunto de regras a serem seguidas.

Os dados nas páginas HTML estão estruturados em formato de uma árvore DOM e portanto é possível fazer a análise sintática das *tags* para extrair informação, geralmente as páginas que são bem estruturadas apresentam um conjunto de dados do mesmo tipo, estruturando em tabelas ou *containers* que rotulam o dado de interesse.

Neste trabalho os dados são extraídos da forma descrita acima. Sabe-se por estudo prévio das páginas quais *tags* que guardam informações relevantes sobre as obras de cada repositório e então é feito a análise dos dados somente naquela sub-árvore de interesse. Esta técnica é eficiente no sentido de limitar o escopo de busca por dados em uma página porém é muito sensível a mudanças que possam ser feitas nas páginas e portanto não é uma estratégia escalável e se mostra mais interessante de ser aplicada quando o domínio é muito específico que é o caso desta obra.

3 TRABALHOS RELACIONADOS

Neste capítulo será feito o detalhamento de informações sobre ferramentas que possuem alguma semelhança com a ferramenta produzida nesta obra, chamada de ECL (Extensor do Currículo Lattes), as diferenças entre elas e suas características únicas. Não foi encontrada nenhuma ferramenta que se especialize em comparar dados de múltiplas fontes com o currículo lattes e armazenar estes dados em um banco de dados. As ferramentas que serão descritas a seguir focam na busca de novos dados em múltiplas fontes.

3.1 SISOB

O SiSOB é uma ferramenta que se propõe a coletar, extrair e persistir informações advindas de diversas fontes da web sobre pesquisadores e agrupar tudo em um banco de dados que permita uma visão unificada dos dados de todos os pesquisadores, forneça análises estatísticas e também permita a manipulação dos dados para que os interessados possam usá-los para outros fins [8].

Este trabalho, diferentemente do ECL utiliza um *crawler* genérico para coletar os dados que tem como escopo toda a *World Wide Web* e é composto por cinco módulos como mostra a arquitetura na figura 3.

Figura 3 - Arquitetura do SiSOB

3.1.1 *CRAWLE*R (M1)

Este módulo foi criado com a intenção de procurar pelas páginas pessoais dos pesquisadores na web e necessita de um conjunto de dados de entrada sobre informações do pesquisador como nome, sobrenome, campos de pesquisa, as letras iniciais do nome e opcionalmente a universidade de afiliação, tudo isso salvo em um arquivo CSV. É usado o motor de busca *DuckDuckGo* e através de tentativa e erro usando todas as combinações possíveis de termos de pesquisa foram

descobertos seis padrões de combinações de dados de entrada que são mais efetivos em buscar resultados interessantes. Alguns são mais restritivos que os outros e nenhum deles possui 100% de acurácia em pesquisar somente dados relevantes, um tipo de padrão possui a combinação dos seguintes metadados: Nome, Sobrenome, Iniciais do nome, sendo que a pesquisa deve ser feita tendo como alcance toda internet.

Exemplo: Carina F. Dorneles

Após a escolha do padrão que será utilizado é feita a pesquisa que irá retornar dois arquivos no formato CSV, um que identifica que nenhuma informação foi encontrada e outro que contém o termo de busca usado e o link contendo a URL que foi encontrada, cada URL servirá de dado de entrada para o módulo dois.

3.1.2 EXTRATOR DE *E-MAI*L (M2)

O módulo serve para extração do e-mail do pesquisador em questão, que servirá de fonte de entrada para o módulo M4 e também para validar a acurácia da página encontrada no módulo M1. Para escolher o e-mail correto correspondente ao pesquisador foi criado um modelo representando a estrutura de vários tipos de e-mails, a partir disto pode se identificar padrões e validar o e-mail correto.

A	B	Z
john.smith@####	jjs@####	webmaster@####
smithjohnc@####	sjj@####	web@####
smith.j.john@####	js@####	mail@####
smith.john@####	sj@####	wmaster@####
johnsmith@####	jjs7r@####	contact@####
smithj####	r7jjs@####	webfeedback@####
smith@####	**jjs**@####	info@####
msmith@####	*jjs*@####	HelpDesk@####
john.s@####	jjs*@####	help@####
johno@####	*jjs@####	desk@####
jjohnsmi@####		*helpdesk*@####
(eight charters)		
jsmith@####		staff@####
smithjj@####		publicaffairs@####
smithj@####		public@####
		public@####
		neurobiology@####
		#department@####
		#university@####
		#field@####
		support@####
		support@####
		news@####

Figura 4 - Modelo de combinações *e-mails* segundo o SiSOB

Como visto na figura 4, os e-mails que possuem combinação de nome e sobrenome são rotulados na categoria “A”, aqueles que possuem combinação das iniciais do nome com números entram na categoria “B” e por fim os que possuem o nome da instituição no e-mail se encaixam na categoria “Z”.

3.1.3 EXTRATOR DE CV E INFORMAÇÕES DA PÁGINA (M3)

O módulo foi dividido em dois pequenos componentes, um para extrair informação do CV e da página e outro que irá procurar por um arquivo contendo o CV do pesquisador não na página atual mas sim vasculhando a internet toda, semelhante ao módulo M1.

O primeiro componente pesquisa por informações do CV e publicações científicas procurando por termos relacionadas como: “cv”, “curriculum”, “vitae”, “pub”, “publications” e se encontrar algo faz o download da página que contém as publicações, por fim pesquisa a página pesquisando por links que terminem com “pdf”, “rdf”, “doc” com a intenção de fazer o download do currículo do pesquisador.

O segundo componente pesquisa o currículo do pesquisador usando padrões semelhantes ao módulo M1 pesquisando por toda a web e salva as URLs que

possivelmente contém o currículo do pesquisador.

3.1.4 PESQUISA POR *E-MAIL* (M4)

O módulo de maneira simplificada serve para enviar e-mails automaticamente aos pesquisadores a partir dos *e-mails* encontrados no módulo M2, são tomadas precauções para evitar problemas como impedir que o e-mail vá para a caixa de spam. A partir disto é enviado um formulário com questões simples para o pesquisador responder e também é requisitado seu currículo para validá-lo com o que foi encontrado no módulo M3.

3.1.5 ANALISADOR E CODIFICADOR DE TEXTO (M5)

O último e mais complexo módulo serve para extrair informação dos currículos e páginas usando os dados de saída do M3 e as respostas das pesquisas enviadas pelo M4, alternativamente pelo fato das saídas de M3 e M4 serem muito variadas e não apresentarem um padrão definitivo elas podem ser complementadas com dados coletados de outros bancos de dados.

Este módulo conta com dois componentes, o primeiro divide os dados de entrada em quatro blocos: dados pessoais, estudos universitários, dados profissionais e publicações. É feita a divisão em quatro blocos para aumentar a precisão no reconhecimento de alguns termos no segundo componente. O segundo componente trabalha com a análise automática dos dados, codificação e extração de informação.

A proposta deste módulo é gerar um arquivo final que contenha informações pessoais, informações sobre as universidades de afiliação, campos de pesquisa, informações profissionais e dados sobre as publicações. Cada bloco contém as seguintes informações:

- Dados pessoais: sexo, nacionalidade, data de nascimento, cidade de nascimento, estado de nascimento e país de nascimento, e-mail, telefone, etc.
- Estudos universitários: áreas de estudo, qualificações, instituição,

cidade, estado e país da instituição e etc

- Dados profissionais: Nome do cargo, duração da atuação, instituição em que o cargo foi exercido, cidade, estado e país e etc.
- Publicações: Todo tipo de publicação acadêmica.

Este arquivo final que é gerado condensa as informações mais relevantes em um currículo que consegue resumir de forma adequada as competências profissionais de um pesquisador. Se comparado ao currículo lattes possui um nível de detalhamento muito menor, uma das razões possivelmente é o fato de que o aumento no nível de detalhes no processo de rotulação dos vários metadados tornaria a ferramenta muito mais complexa visto que fazer isto de forma automatizada é suscetível a erros.

3.2 CITE SEER X

Cite Seer X[6] é uma biblioteca digital que possui um motor de busca que permite o usuário ter acesso a milhões de arquivos sobre as publicações de pesquisadores. Tem seu foco principal na literatura relacionada à computação e ciência da informação, permite o usuário ter acesso a metadados como: título, autores, abstract e outros dados relacionados a publicações e também fazer o download em PDF das publicações.

Assim como o ECL o Cite Seer X possui um *crawler* focado chamado citeseerxbot que vasculhar a web em busca de novos dados para serem indexados. O *crawler* é focado pois ele faz a busca a partir de um arquivo contendo várias URLs de domínios acadêmicos que são adicionadas diariamente de forma incremental. Os usuários também podem contribuir através do site enviando URLs que contenham arquivos em PDF para que o *crawler* extraia os metadados.

A arquitetura do Cite Seer X é dividida em um *front end* e *back end*, no *front end* os dados das buscas feitas pelos usuários são processados por um web *service* que contém três servidores [12]. O *index server* é responsável pelas requisições feitas pelo usuário e por retornar as pesquisas, o *repository server* contém todos os documentos presentes no site e o *database server* contém todos

6http://citeseerx.ist.psu.edu/index

os metadados.

O *back end* é onde se encontra o *crawler* que inicia sua busca através do arquivo de URLs semente extraindo arquivos em PDF, esses arquivos são transferidos para um módulo que irá extrair e classificar metadados dos arquivos, por fim um módulo irá escrever todos os metadados encontrados, no *database server,* salvar todos os arquivos no *repository server* e atualizar os dados indexados no *index server.*

Apesar de o *crawler* ser focado ele precisa indexar todos os dias números gigantescos de páginas diferentes o que torna a tarefa de classificação dos metadados mais complexa visto que há uma grande variedade no formato dos arquivos baixados. São usadas técnicas de IA com a intenção de otimizar a extração de metadados e documentos, quatro técnicas diferentes que tratam de diferentes problemas são usadas nos módulos de extração de metadados e ingestão, mostrados na figura 5.

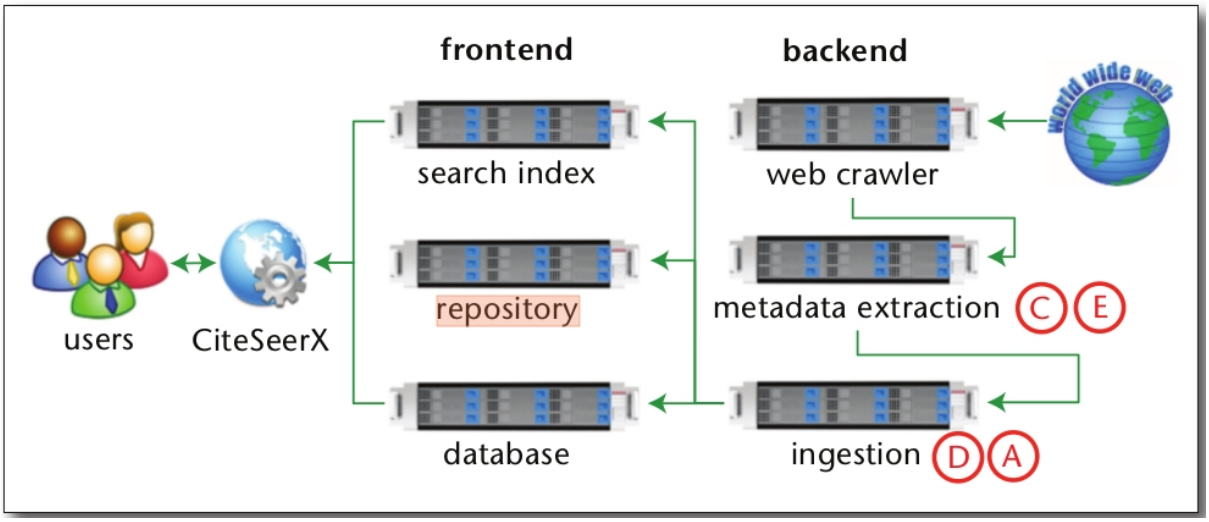


Figura 5 - Arquitetura do Cite Seer X

A figura 5 possui quatro *tags* que são A, C, D e E, cada uma delas representa uma tarefa que será resolvida com a ajuda de um algoritmo de IA.

Tag A representa a tarefa de tentar diferenciar o nome de autores que possuem nomes parecidos ou identificar o nome exato dos autores envolvidos separando os sobrenomes e nomes adequadamente. *Tag* C é usada para separar os documentos acadêmicos dos não acadêmicos. *Tag* D elimina réplicas de

documentos ou documentos muito semelhantes entre si. *Tag* E é usada para extrair os metadados das publicações.

3.2.1 CLASSIFICAÇÃO DE DOCUMENTOS

Esse módulo serve para classificar os documentos como acadêmicos ou não e usa um filtro com um modelo baseado em regras que procura identificar keywords como "*references"*, "*bibliography"* e suas variantes na seção de referências dos arquivos. Foram testados vários algoritmos como: *Support Vector Machine*(SVM), regressão logística e o classificador *Naive-Bayes* e o primeiro apresentou os melhores resultados com precisão de 90% ao classificar os arquivos científicos corretamente.

3.2.2 EXTRAÇÃO DE METADADOS

Os metadados são extraídos dos arquivos PDF encontrados pelo classificador de documentos e possuem 15 campos: título, autores, afiliação, endereço, nota, e-mail, data, abstract, introdução, telefone, palavras chave, web, grau de ensino e informação da página. A classificação é feita usando SVM e tem como objetivo classificar os textos em cada classe correspondente a um campo citado acima, o extrator conseguiu uma taxa de acurácia de 92.9% usando-se um conjunto de dados de teste de 935 arquivos de publicações na literatura de ciências da computação.

3.2.3 DUPLICAÇÃO DE DOCUMENTOS

Apesar de ser algo muito raro, as vezes pode acontecer de bibliotecas digitais ou páginas pessoais conterem documentos duplicados ou extremamente parecidos. Documentos que são idênticos são fáceis de se identificar pois possuem todos os bits idênticos e geram o mesmo valor de hash. Já os arquivos semelhantes são mais complexos e são tratados através de um algoritmo chamado *key-mapping* que utiliza como entrada os dados dos autores e títulos que são presentes em quase todos os arquivos.

3.2.4 EXTRATOR DE TABELAS

Muitos artigos científicos possuem tabelas que apresentam dados valiosos sobre estatísticas e amostragem de resultados, porém são difíceis de extrair de forma automática por não possuírem um padrão bem definido. O extrator divide os componentes literais do arquivo em várias *boxes* e tenta procurar por palavras como "*TABLE"*, "*table"* e palavras relacionadas, se identificar alguma palavra de interesse significa que existe uma chance de um componente adjacente ser uma tabela.

3.3 COMPARAÇÃO ENTRE OS TRABALHOS

Nas seções acima foram descritos dois diferentes *crawlers*, sendo um deles muito conhecido e usado pela comunidade científica (*Cite Seer X*) e a seguir serão feitas comparações entre eles para mostrar as diferenças e semelhanças com o *crawler* desenvolvido neste trabalho.

Vale ressaltar que as duas ferramentas desenvolvidas nas obras descritas acima tem um escopo muito maior e são usadas em ampla escala, já o ECL possui um escopo muito menor e limitado e tem seu foco em cima do currículo lattes por ser bastante conceituado na comunidade científica brasileira.

Característica	*Cite Seer X*	SiSOB	ECL
Tipo de *crawler*	Focado	Genérico	Focado
Navegação entre páginas	Sim	Sim	Sim
Uso de técnicas de aprendizagem de máquina	Sim	Não	Não
Entrada manual de dados	Não	Sim	Sim
Extração de dados de publicações científicas	Sim	Sim	Sim
Objetivos	Motor de busca que indexa milhões de obras científicas	Explorar mobilidade dos pesquisadores através do CV e produzir estatísticas sobre os dados	Comparar dados de fontes externas ao lattes para incrementar o currículo do pesquisador

Tabela 1 - Comparação entre o ECL e os trabalhos relacionados

Apesar dos objetivos das ferramentas citadas na tabela 1 serem diferentes, elas possuem muitas coisas em comum, por exemplo, todas extraem dados de publicações científicas e também possuem a capacidade de navegar entre as páginas.

As duas ferramentas analisadas acima possuem um escopo muito maior que o ECL, o SiSOB vasculha a *web* inteira enquanto o *Cite Seer X* apesar de ser um *crawler* focado igual ao desta obra faz uma busca por dados muito mais extensa. O Cite Seer X foi desenvolvido em 1997 e continua sendo aperfeiçoado até o presente momento e conta com a ajuda de colaboradores de todo o mundo por ser um projeto *open-source*, porém o ECL tem seu foco no currículo lattes que é mais bem difundido nacionalmente.

Uma das vantagens do ECL é por ele ser muito especializado e pesquisar somente os repositórios da DBLP, GoogleScholar e ResearchGate. Os resultados retornados são muito mais satisfatórios do ponto de vista que as fontes foram estudadas previamente e são bastante reconhecidas pela sua qualidade no mundo acadêmico, também com um menor escopo é possível estudar as páginas e tratar os detalhes de cada uma individualmente.

4 ECL

Neste capítulo será detalhado o funcionamento do *crawler* ECL (Extensor do Currículo Lattes) que irá acessar as publicações de três repositórios diferentes e comparar seus dados com as publicações do currículo lattes. Ao longo do capítulo será mostrada a arquitetura do sistema e os detalhes de implementação de todos os módulos envolvidos.

4.1 VISÃO GERAL

O *crawler* possui uma arquitetura simples que pode ser dividida em quatro partes:

1 - Conexão com um servidor de *proxy* que irá intermediar as requisições enviadas aos repositórios do GoogleScholar e ResearchGate.

2 - Acesso aos repositórios e extração dos dados das publicações.

3 - Análise sintática do currículo lattes para comparação com os dados extraídos dos repositórios.

4 - Persistência no banco de dados do currículo lattes usado para comparação e das publicações que o *crawler* julgar que não estão contidas no lattes.

Para acessar os currículos lattes é necessário resolver um *captcha* e tentar resolvê-lo seria um esforço que foge do escopo deste trabalho, devido a isto, para que a ferramenta funcione adequadamente é necessário que o *download* do currículo lattes seja feito manualmente em formato XML e o arquivo seja colocado em uma pasta específica que será lida pelo *crawler* em busca dos currículos para fazer a análise sintática. A arquitetura do *crawler* ECL pode ser vista na fgura 6 que mostra os módulos envolvidos no sistema.

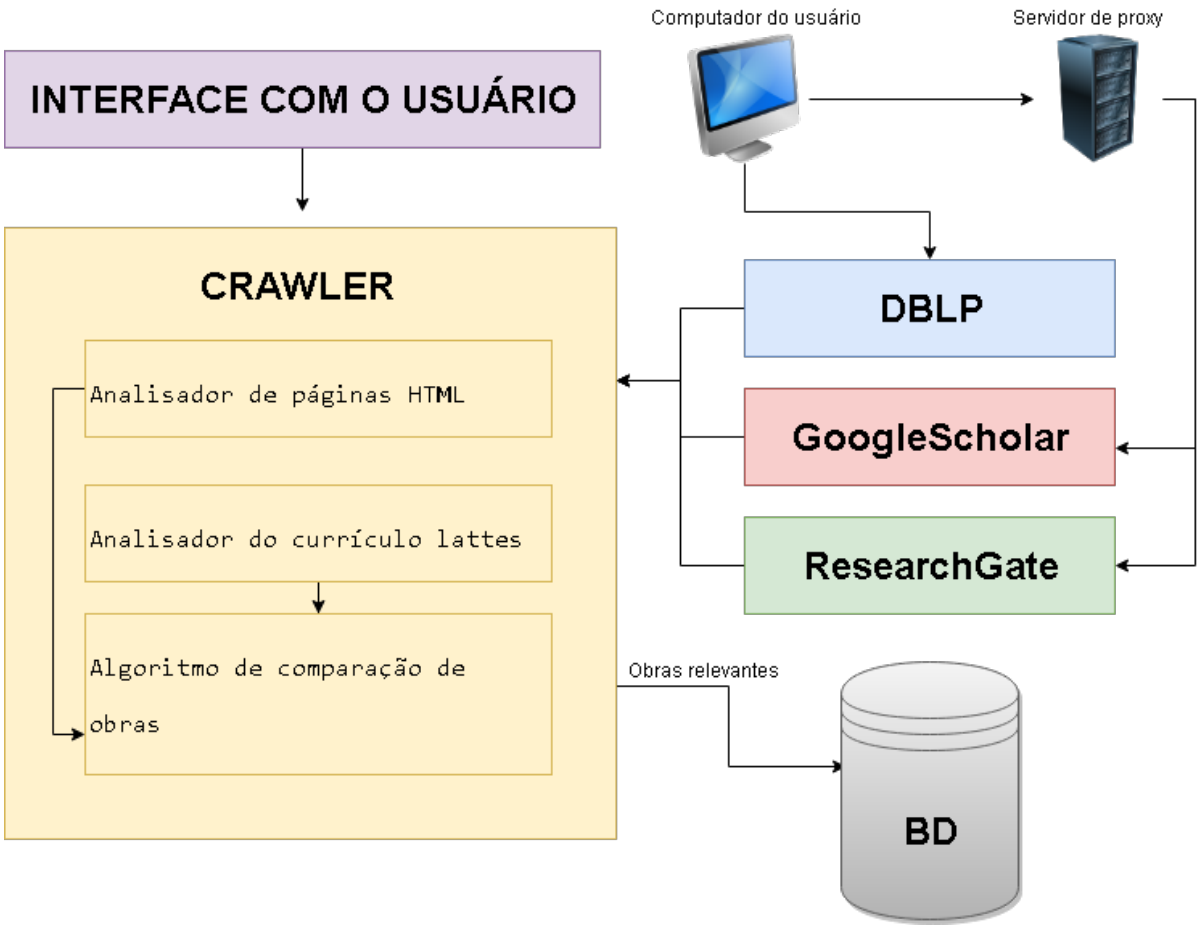


Figura 6 - Arquitetura do ECL

4.1.1 CONEXÃO COM UM SERVIDOR DE PROXY

Um servidor de *proxy* age como um intermediário entre as requisições enviadas pelos usuários e o servidor que se deseja acessar [9]. Ao longo do desenvolvimento do *crawler* notou-se a necessidade de criar este módulo pois os repositórios do GoogleScholar e ResearchGate limitam o acesso aos seus recursos para evitar sobrecarga no servidor. Mesmo tentando acessar estes dois repositórios impondo um limite na velocidade em que o *crawler* faz acesso aos recursos, após um determinado período de requisições o acesso era bloqueado via endereço de IP resultando em erros HTTP como: 429 Too many requests ou 404 Not Found.

Através de testes realizados de forma empírica percebeu-se que uma solução para o problema seria rotacionar entre vários endereços de IP impedindo que um mesmo IP faça mais requisições que o permitido e seja bloqueado [13] e através da conexão com o servidor de proxy é possível alterar o IP nas requisições.

4.1.2 ACESSO AOS REPOSITÓRIOS E EXTRAÇÃO DOS DADOS DAS PUBLICAÇÕES

Nesta parte o *crawler* faz o acesso aos repositórios da DBLP - *Computer Science Bibliography*, GoogleScholar e ResearchGate. Estes repositórios foram escolhidos por apresentarem características diferentes entre si, por exemplo, na DBLP os dados de produções adicionadas são obtidos pela editora de um volume ou por um organizador de um evento e são adicionados manualmente ao site, as produções do GoogleScholar são indexadas no site através de um *crawler* que vasculha a web sendo assim menos confiáveis e por último as publicações do ResearchGate são adicionadas manualmente pelos próprios pesquisadores o que torna os três repositórios distintos entre si na forma como coletam os dados.

A DBLP abrange apenas produções da literatura de Ciências da Computação, já o GoogleScholar e o ResearchGate abrangem publicações de diversas áreas do conhecimento.

Cada repositório possui características diferentes em relação ao HTML e nesta parte o *crawler* precisa fazer a análise sintática e tratar os dados de forma adequada, por exemplo, é esperado que o campo “year” de uma publicação contenha um número de quatro dígitos e não possua letras mas não se pode assumir que o dado neste campo venha sempre formatado da maneira esperada.

4.1.3 ANÁLISE SINTÁTICA DO CURRÍCULO LATTES E COMPARAÇÃO COM OS DADOS EXTRAÍDOS DOS REPOSITÓRIOS

Aqui os metadados relevantes do currículo lattes de cada professor são extraídos para poder comparar com os metadados dos outros repositórios, é feita uma comparação de todos com todos usando um algoritmo que foi criado através de testes empíricos e é usado uma métrica de similaridade de strings para comparar se os dados são ou não semelhantes, os detalhes sobre como esta comparação é feita e que métricas foram usadas serão detalhados no capítulo de implementação.

4.1.4 PERSISTÊNCIA NO BANCO DE DADOS DAS OBRAS RECUPERADAS

Este módulo trata de persistir todas as publicações que foram comparadas com as publicações do lattes e foram identificadas como novas obras, ou seja, que não estão contidas no lattes, o próprio currículo lattes também é armazenado para se ter conhecimento da fonte original de dados que foi usada como base para comparação.

4.2 IMPLEMENTAÇÃO

O *crawler* 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:

1. Jsoup[7] é uma biblioteca em JAVA desenvolvida para trabalhar com páginas HTML e possui uma estrutura para manipulação e extração de dados.
2. Java-string-similarity[8] é uma biblioteca que implementa diversas métricas de similaridade e distância de *strings.*
3. PostgreSQL JDBC Driver[9] é a biblioteca usada para permitir que o programa em JAVA se conecte com o banco de dados PostgreSQL.

O projeto contém duas pastas que são nomeadas como “lattes” e “dblp”. Na pasta lattes devem ser salvos pelo usuário o currículo lattes de todos os pesquisadores que terão suas obras analisadas pelo *crawler*, na pasta “dblp” são salvos todos os arquivos em XML que são baixados pelo *crawler* e contém o

7https://jsoup.org/
8https://github.com/tdebatty/java-string-similarity

9https://jdbc.postgresql.org/

conjunto de obras dos pesquisadores. O *crawler* possui o seguinte diagrama de classes apresentado na figura 7.

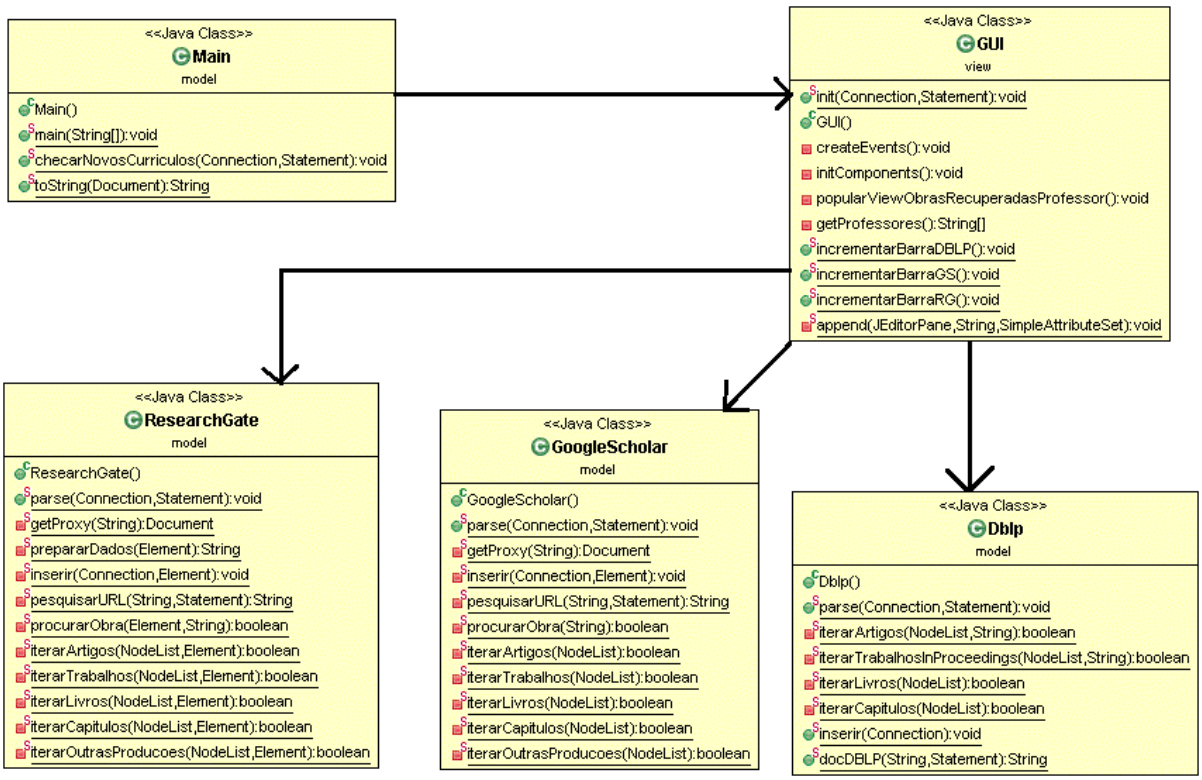


Figura 7 - Diagrama de classes da aplicação

A classe Main faz a conexão com o banco de dados e por isso guarda em seus atributos dados como URL, usuário e senha. Após se conectar ao banco de dados é executado o método “checarNovosCurriculos” que tem o intuito de atualizar a tabela “professores”, criando e salvando um id, nome e o currículo lattes em XML de acordo com os currículos lattes contidos na pasta “lattes”.

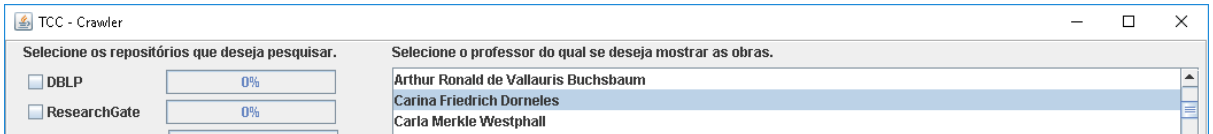
O algoritmo primeiramente gera o id que será utilizado caso necessite adicionar um novo registro a tabela professores, após isso itera sobre todos os currículos lattes contido na pasta “lattes”, se o nome do pesquisador contido na tag “NOME-COMPLETO” não estiver contido em nenhuma instância da tabela professores então é criada uma nova instância com o id, nome do professor e currículo na tabela. O algoritmo pode ser visto com mais detalhes na figura 8.

```
BEGIN checarNovosCurriculos(Connection conn, Statement myStmt)
        File folder = new File("files/lattes/");
        File[] listOfFiles = folder.listFiles();
        ResultSet myRs;
        int id = 0;
        myRs = myStmt.executeQuery("SELECT * FROM professores\n" +
                    "WHERE id = (\n" +
                    "    SELECT MAX(id) FROM professores)");
       IF(myRs.next())
                id = myRs.getInt("id");
       ENDIF
        FOREACH (listOfFiles) {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            org.w3c.dom.Document lattes = dBuilder.parse(new File("files/lattes/" + listOfFiles[k].getName()));
            NodeList curriculo = lattes.getElementsByTagName("CURRICULO-VITAE");
            NodeList dadosGerais = ((Element)curriculo.item(0)).getElementsByTagName("DADOS-GERAIS");
            Node rootLattes = dadosGerais.item(0);
            Element l = (Element) rootLattes;
            String nome = l.getAttribute("NOME-COMPLETO");
            myRs = myStmt.executeQuery("SELECT nome FROM professores WHERE nome = "+ "'"+nome+"'");
            IF(!myRs.next())
                id++;
                PreparedStatement stmt = conn.prepareStatement("INSERT INTO professores (id, nome, lattes) VALUES (?, ?, XML(?))");
                stmt.setInt(1, id);
                stmt.setString(2,  nome);
                stmt.setString(3,  toString(lattes));
                stmt.executeUpdate();
                conn.commit();
            END IF
        END FOR
END METHOD
```

Figura 8 - Demonstração em pseudo-código da atualização da tabela professores

Antes de invocar os métodos que cuidam da extração de dados das fontes é feito um DELETE dos dados das tabelas que contém as produções, a razão para isto se deve ao fato de que é mais eficiente apenas inserir novos dados na tabela ao invés de checar se a produção já foi armazenada e então inseri-los. Evitar a iteração sobre todas as obras das fontes de dados para checar se ela já foi armazenada tem o ponto positivo de evitar *overhead* tornando a execução da aplicação mais rápida.

Ao final da execução do algoritmo descrito acima é invocado um método que inicializa a classe GUI, mostrada na figura 9, que é responsável por apresentar a interface de usuário.



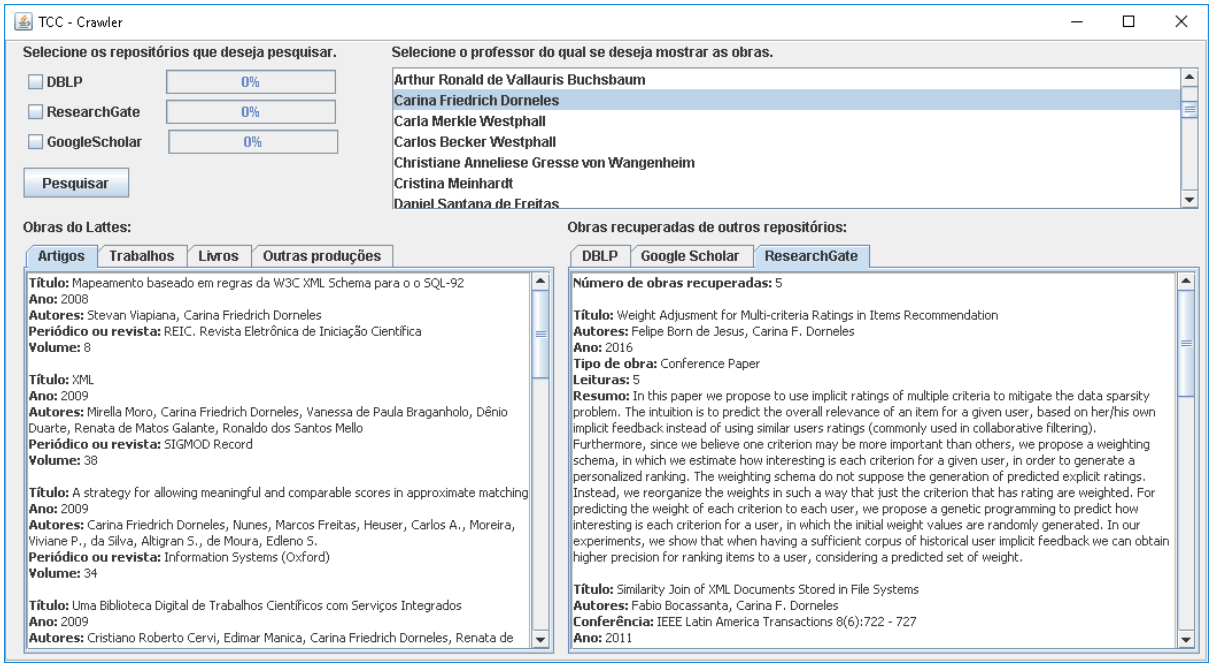


Figura 9 - Interface gráfica do ECL mostrando uma visão unificada das obras recuperadas e das obras do lattes

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 o banco de dados inicia uma busca por todas as obras recuperadas e também todas as obras do lattes do professor 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.

Apesar de o projeto possuir uma classe para cada repositório de dados o algoritmo usado nelas é extremamente semelhante e só difere em pequenos detalhes referentes a particularidades de cada página. A seguir será explicado o algoritmo que é usado em todas as classes e depois será detalhado a particularidade de cada uma.

4.2.1 EXTRAÇÃO E COMPARAÇÃO DE DADOS

O algoritmo para extração e comparação de dados é semelhante em todas as classes, é necessário iterar por todos os currículos lattes contidos na pasta “lattes” e

extrair apenas os dados relevantes.

O *crawler* inicialmente faz a análise sintática dos dados contidos no currículo e armazena em uma variável que contém a árvore DOM e permite o acesso aos dados para saber que URL será acessada e também qual o id do pesquisador que será salvo na coluna “id_prof” para fazer referência ao pesquisador.

É feita uma pesquisa no banco de dados na tabela “professores” pesquisando pelo nome do pesquisador, este nome é extraído acessando a árvore pelas *tags* "CURRICULO-VITAE" > "DADOS-GERAIS" > "NOME-COMPLETO", se a consulta retornar algum valor é extraído o link de uma das colunas (linkdblp, linkgooglescholar, linkresearchgate) que contém a URL das obras do pesquisador naquele repositório e também o id. As colunas contendo os *links* de acesso aos repositórios foram populadas manualmente, o *crawler* no presente momento não busca a URL de forma automatizada.

A seguir, um exemplo de consulta descrita acima que é realizada com o seguinte comando SQL:

“ SELECT linkdblp, id FROM Professores WHERE nome LIKE ‘Arthur’ ”

Se uma URL for encontrada então é feita a análise sintática de todas as *tags* que contêm as publicações do pesquisadores, estas *tags* são:

1. "TRABALHO-EM-EVENTOS": Contém dados sobre todos os trabalhos completos e resumos publicados em anais de congressos.
2. “ARTIGO-PUBLICADO”: Contém dados sobre todos os artigos publicados em periódicos.
3. “LIVRO-PUBLICADO-OU-ORGANIZADO”: Contém dados sobre todos os livros publicados/organizados ou edições.
4. “CAPITULO-DE-LIVRO-PUBLICADO”: Contém dados sobre todos os capítulos de livros publicados.
5. “OUTRA-PRODUCAO-BIBLIOGRAFICA”: Contém dados de outras produções bibliográficas que não se encaixam em nenhuma das categorias descritas acima.

A extração dos dados das páginas funciona do mesmo modo no caso do GoogleScholar e ResearchGate em que o acesso aos dados é feito através das tags HTML, os dados das obras são estruturados em forma de árvore e estudando cada página individualmente é possível saber qual metadado está contido dentro de cada

tag. Esta abordagem é mais fácil de programar por não necessitar de nenhuma técnica avançada como o uso de IA para identificar os metadados nas páginas. Tem como desvantagem o fato de ser inflexível pois se a estrutura das *tags* for mudada na página o *crawler* não vai conseguir identificar onde se encontram os elementos e será necessário uma intervenção manual para descobrir novamente como funciona a estrutura da árvore DOM da página.

Para a DBLP não é necessário acessar as tags da página em HTML pois este repositório permite o acesso dos dados estruturados em XML assim como o currículo lattes. A justificativa se da devido à facilidade de trabalhar com o XML e também é evitado o problema de precisar reescrever o código caso a página mude sua estrutura.

É feito o *download* dos dados em XML e então é feita a análise sintática, antes de ser feito o *download* é chamada uma função que verifica se o arquivo já foi baixado, se sim o nome do arquivo é retornado para ser feita a análise sintática posteriormente, se não pesquisa no banco de dados pelo *link* usando o nome do professor, se possuir um *link* válido então o arquivo é criado senão o algoritmo busca outro professor para fazer a análise.

Com os dados estruturados na memória é feito uma comparação de todos com todos para identificar quais dados são diferentes e devem ser persistidos, por exemplo, iterar sobre todas as obras da DBLP verificando se a obra é semelhante a alguma obra do lattes iterando por todas as obras do lattes contidas nas tags descritas que contém publicações.

```
1 FOR EACH(obraDBLP)
2    FOR EACH(obraLattes)
3       IF(!compararObras(obraDBLP, obraLattes))
4          persistirObraNoBanco(obraDBLP);
5       END IF
6    END FOR
7 END FOR
```

Figura 10 - Pseudo-código ilustrando o algoritmo descrito acima

O que muda entre os repositórios é o modo como são extraídos os metadados e o tratamento deles para evitar exceções no código.

O algoritmo usado para comparar os metadados entre as fontes tem como entrada metadados semelhantes das duas fontes comparadas como: título, ano e

autores, páginas, volume, sendo que os dois últimos só são usados pela DBLP que é a única fonte que apresenta esses dados de forma bem estruturada.

A comparação é feita através do uso de algoritmo de similaridade entre *strings*, foi usado o algoritmo LCS (*Longest Common Subsequence*) que consiste em achar a maior subsequência comum entre duas ou mais sequências, também é usado pela ferramenta *diff utility* usada pelo Git para conciliar as diferenças entre dois repositórios de código e por isso foi utilizado neste trabalho. Tem como entrada duas *strings* e sua saída é um valor normalizado entre 0 e 1, sendo que 0 representa duas *strings* iguais e 1 totalmente diferentes.

O título das fontes de dados é transformado em minúsculo para todas as letras, já que neste caso letras maiúsculas e minúsculas não fazem diferença para fins de comparação mas fariam o algoritmo considerar o caractere como diferente.

Como mostra o código na figura 11, é feita uma comparação do título nas duas fontes de dados usando o LCS, se o resultado retornar um valor menor que 0.1 então as obras são semelhantes e itera na próxima obra. Em caso negativo é feita a checagem se o valor é menor que 0.3 e se for é comparado o ano das duas obras, se os anos forem iguais então é assumido que a obra é a mesma e itera na próxima obra. Por fim é checado se o valor é < 0.9 então para que as obras sejam consideradas semelhantes é preciso que o ano seja igual e todos os autores das duas obras também, isto é feito pois mesmo que o título seja muito diferente pode acontecer de um título ser resumido e o outro completo tornando as *strings* muito diferentes entre si. Também é possível que mesmo que o título esteja escrito em inglês em uma fonte e em português na outra as obras sejam consideradas a mesma.

Se for feita a comparação de uma obra de outros repositórios com todas as obras do lattes e não for encontrada semelhança nenhuma então a obra é salva no banco de dados, como cada repositório possui características diferentes são salvos diferentes tipos de metadados para cada um deles.

```
BEGIN compararObras(obraDBLP, obraLattes)
obraDBLP.tituloDBLP.tolowerCase();
obraLattes.tituloLattes.tolowerCase();
IF(lcsdistance(obraDBLP.tituloDBLP, obraLattes.tituloLattes) < 0.1)
    return TRUE; #As obras são semelhantes logo não precisa armazenar nada
END IF

IF(lcsdistance(obraDBLP.tituloDBLP, obraLattes.tituloLattes) < 0.3)
    IF(obraDBLP.anoDBLP == obraLattes.anoLattes)
        return TRUE; #As obras são semelhantes logo não precisa armazenar nada
    END IF
END IF

IF(lcsdistance(obraDBLP.tituloDBLP, obraLattes.tituloLattes) < 0.9)
     IF(obraDBLP.anoDBLP == obraLattes.anoLattes)
        IF(obraDBLP.autoresDBLP == obraLattes.autoresLattes)
           return TRUE; #As obras são semelhantes logo não precisa armazenar nada
        END IF
END IF

return FALSE; # Se chegou aqui as obras são diferentes logo é necessário armazenar os dados
END METHOD
```

Figura 11 - Pseudo-código ilustrando o algoritmo de comparação de dados dentro do loop que itera sobre todas as obras

4.2.2 ESTRUTURA DAS FONTES DE DADOS

Cada repositório possui suas particularidades em relação a estrutura de dados e também quanto aos dados que são fornecidos. Assim foi necessário o estudo e entendimento de cada um deles, abaixo será detalhada a estrutura de cada um dos repositórios.

4.2.2.1 DBLP

O DBLP é o repositório em que os dados estão melhores estruturados por fornecer a possibilidade ao usuário de acessar um documento em XML que contém as obras em uma estrutura padrão.

Existem dois tipos de dados, dados pessoais do pesquisador e dados sobre as obras. A primeira tag “person” armazena dados pessoais do pesquisador que não serão relevantes para o *crawler* já que a intenção é comparar as obras entre si.

As obras são armazenadas dentro das tags “r” e armazenam dados sobre as publicações dos pesquisadores e apresentam vários tipos possíveis de tags como: “*article*”, “*inproceedings*”, “*proceedings*”, “*book*”, “incollection”, “phdthesis”, “mastersthesis” e “www”. É feito a análise sintática somente das quatro primeira

tags citadas anteriormente pois elas contém as informações relevantes de obras para o *crawler*.

1. *article*: Contém dados sobre um artigo publicado em revistas científicas.
2. inproceedings: Contém dados de um *paper* publicado em conferências ou *workshops*.
3. proceedings: Contém dados sobre os volumes das *proceedings* de uma conferência ou *workshop.*
4. *book*: Uma monografia ou edição de coleções de artigos.

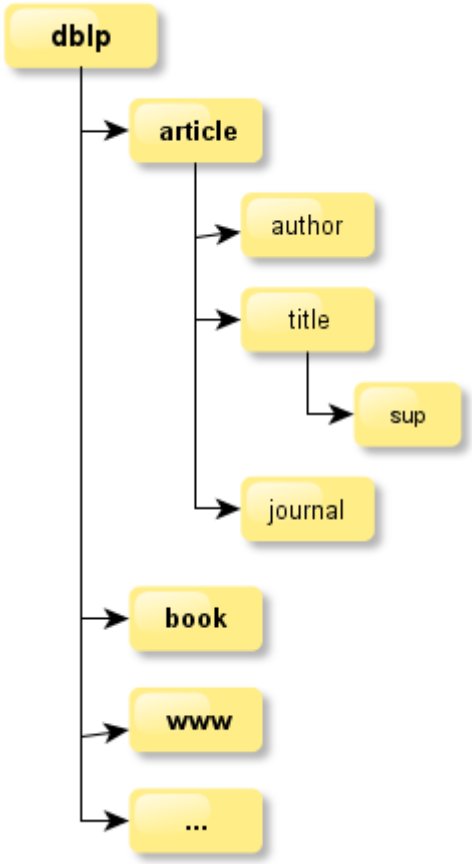


Figura 12 - Esquemático da estrutura do arquivo XML da DBLP

Todos os tipos de obras contém quatro *tags* essenciais que contém metadados que são usados na comparação entre obras.

1. “*title*”: Título da obra.
2. “year”: Ano da obra.
3. “authors”: Autores da obra.

4. “editors”: Editores da obra.

As outras *tags* possuem informações que não são relevantes para a comparação mas que possuem metadados sobre a obra que são armazenados, como são usados quatro tipo de obras diferentes classificadas pela DBLP os metadados contidos nelas também variam, estes metadados são:

1. “*pages*”: Indica a quantidade de páginas da obra informando o número da página inicial e possivelmente o número da página final. Exemplo: (50-70).
2. “*booktitle”*: Nome da conferência em que a obra foi publicada.
3. “*journal*”: Nome do *journal* em que a obra foi publicada.
4. “volume”: Número do volume de um *journal*.
5. “*number*”: Número de edição.
6. “*crossref*”: A chave de uma entrada *cross-reference*.
7. “ee”: Contém a URL de um site que contém dados da obra.
8. “url”: Especifica um *hyperlink* local relativo a *homepage* da DBLP.
9. “*key*”: Chave única da DBLP desta obra.
10. “mdate”: Última data em que a obra foi modificada.

Vale ressaltar que apesar de a estrutura XML ser bem estruturada os metadados contidos nas *tags* tem que ser extraídos com cautela, por exemplo, na tag “*year*” é esperado que sempre armazene um valor de ano de quatro dígitos mas isto nem sempre acontece e o código trata esse tipo de exceção.

4.2.2.2 GOOGLESCHOLAR

Neste repositório os dados estão estruturados na página no formato HTML, é necessário navegar por *tags* específicas que guardam informações de cada obra, diferentemente da DBLP as obras não possuem nenhum tipo de *tag* que identifique a natureza da mesma.

A biblioteca Jsoup permite guardar em uma lista o conteúdo da página e sua estrutura. Analisando a página pode ser visto que todas as obras estão contidas dentro da *tag* “gsc_a_tr” e com o uso do método *getElementsByClass(String* nomeClasse*)* é possível armazenar todos os elementos que contenham o atributo *class*.

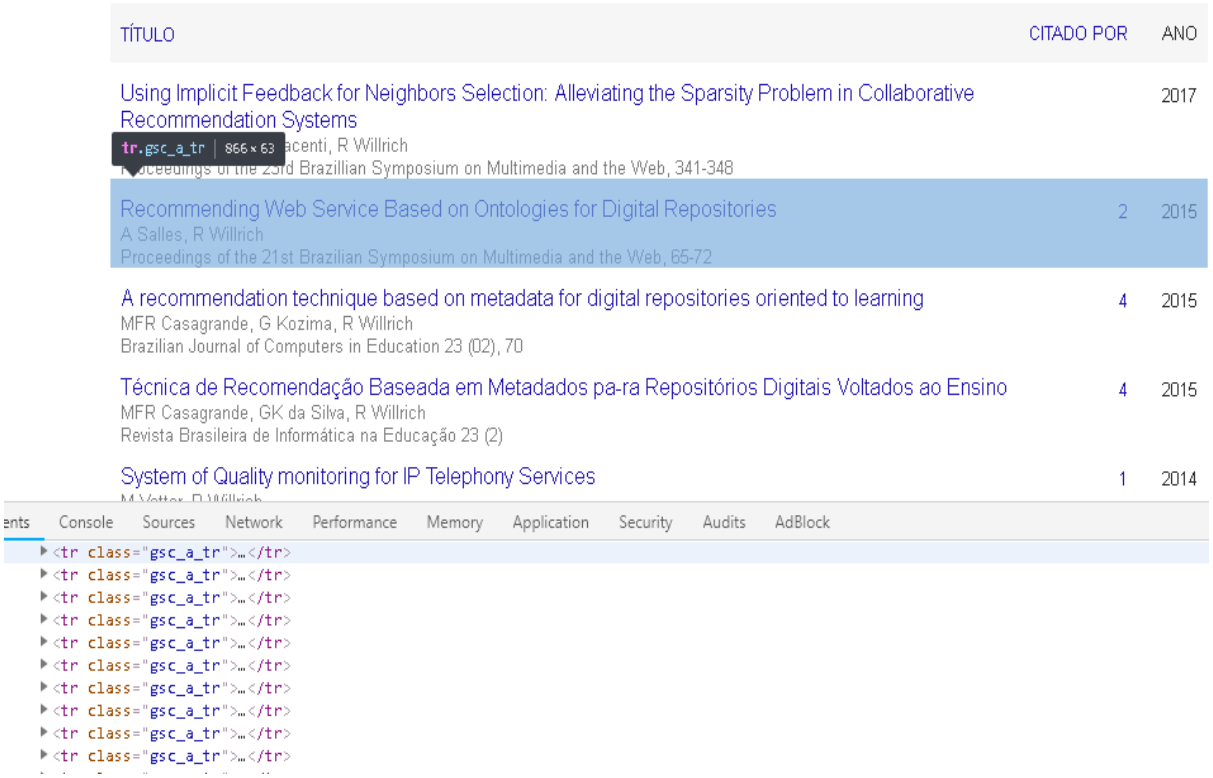


Figura 13 - *Tag* que armazena os metadados das obras no GoogleScholar

Dentro da *tag* “gsc_a_tr” existem mais três *tags*:

1. “gsc_a_t”: Possui metadados separados sobre o título da obra, autores , evento, conferência e universidade dependendo de tipo da natureza da obra.
2. “gsc_a_c”: Número de citações da obra.
3. “gsc_a_y”: Ano da obra.

Foi necessário a criação um algoritmo para paginação visto que cada página apresenta no máximo 100 resultados de obras, um usuário que usa o *browser* precisa clicar em um botão com a *label* “MOSTRAR MAIS” e através da execução de código *JavaScript* é 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.

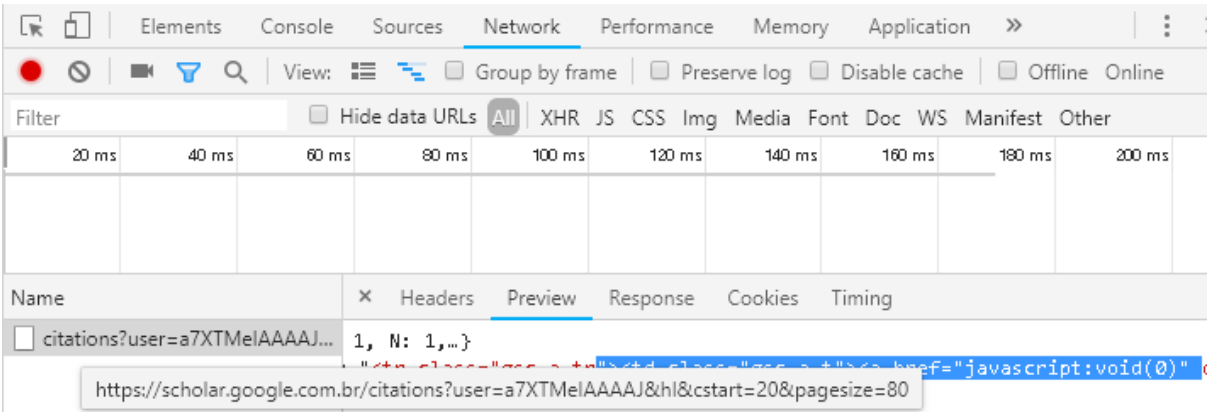


Figura 14 - Padrão identificado na URL, através do estudo do *log* da rede

É possível identificar que ao clicar no botão “MOSTRAR MAIS” é feita uma requisição para uma URL que irá carregar novos dados na página, como pode ser visto na figura 14 esta URL possui um simples padrão de concatenação da URL da página do pesquisador + “&cstart=X&pagesize=Y”. X representa o valor da obra inicial que será consultada e Y o número de obras a serem carregadas que terá o valor fixo de 100 pois só podem ser carregadas 100 obras por requisição. Caso seja feita uma requisição com valores que não coincidem com o número de obras a página mostra que não existe nenhum artigo no perfil e não mostra uma *label* que indica a quantidade de artigos mostrados na página e isto é usado como critério de parada do *crawler*.

No exemplo ilustrado na figura 15 é feita a requisição pela URL: “https://scholar.google.com.br/citations?user=a7XTMeIAAAAJ&hl&cstart=700&pagesize=100” porém o pesquisador não possui 800 obras no seu perfil e a página não retorna artigo nenhum.

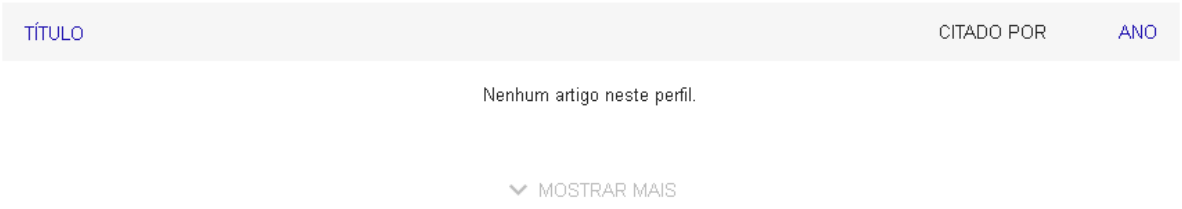


Figura 15 - Exemplo de acesso a URL com paginação incorreta

Figura 16 - Exemplo de acesso a URL com paginação correta

Na ilustração da figura 16 é possível observar que existe um componente que mostra quantos artigos são mostrados por página e este componente é acessado pelo método doc.getElementById(*tag*) utilizando a *tag* “gsc_lwp”.

O algoritmo inicializa o X com o valor 0 e Y com valor 100 para os parâmetros de “&cstart=X&pagesize=Y”, se o componente com a *tag* “gsc_lwp” conter no número após a *string* “-” um valor menor que X, então não existe mais artigos nesta página e encerra a pesquisa para este pesquisador, se o valor for igual ao de X o valor de X é incrementado em 100 e a execução continua até que o número de artigos seja menor que X ou a página não possuir o componente.

```
1  int X = 0;
2  String URL = URLpaginaProfessor;
3  String page = "&cstart=" + X + "&pagesize=100";
4  doc = Jsoup.connect(url+page);
5  WHILE (doc.getElementById("gsc_lwp").exists() && doc.getElementById("gsc_lwp").numeroDeArtigos < X)
6  algoritmoParseComparacaoDeDados();
7  X += 100;
8  page = "&cstart=" + X + "&pagesize=100";
9  doc = Jsoup.connect(url+page);
10 END WHILE
```

Figura 17 - Pseudo-código do algoritmo de paginação

É necessário verificar se o componente existe pois pode acontecer o caso de a expressão (doc.getElementById("gsc_lwp").numeroDeArtigos == X) for verdadeira e não existir mais nenhum artigo para ser pesquisado então o algoritmo identifica que é preciso sair do *while* através da não existência do componente, sem isso o código poderia apresentar erro por tentar acessar dado de um componente que não existe.

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.

Figura 18 - Ilustração de dados que aparecem incompletos

Para resolver este problema é necessário extrair um *link* do atributo “data-href” que está contido dentro da tag “gsc_a_t” como mostrado na figura 18, concatenando a string da *homepage* 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.

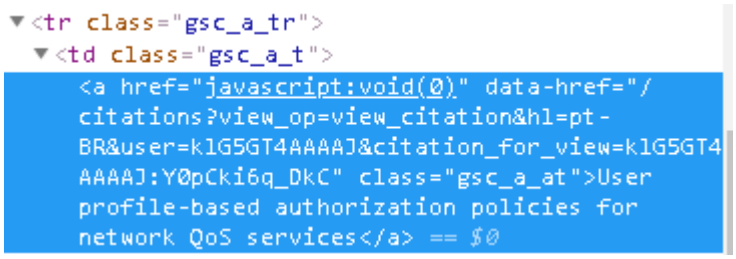


Figura 19 - Atributo “data-href” contendo um *link* para acesso a uma página

4.2.2.3 RESEARCHGATE

O repositório do ResearchGate semelhantemente 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 de *tags* HTML. Usando o método doc.getElementsByClass(*tag*) é possível acessar todas as obras que estão contidas na *tag* que possui a seguinte *label*: "nova-v-publication-item__stack nova-v-publication-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 a *tag* com a *label* “nova-v-publication-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 da *tag* 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. Os três primeiros metadados são usados para comparar as obras do lattes e ResearchGate e caso o *crawler* identifique que as obras são distintas é feito o acesso a URL contida no campo “href” da mesma *tag* que contém o título da obra, o último metadado que identifica o tipo da obra é salvo para posteriormente ser persistido no banco.

```
doc.Jsoup.connect(url);
doc.getElementsByClass("nova-v-publication-item__stack nova-v-publication-item__stack--gutter-m");
FOR EACH(obraResearchGate)
    FOR EACH(obraLattes)
        IF(obraResearchGate == obraLattes)
            doc.Jsoup.connect(doc.getNovaURL()); #Faz acesso a nova URL contida no HTML
        END IF
    END FOR
END FOR
```

Figura 20 - Pseudo-código ilustrando o algoritmo que visita a página com mais detalhes referentes à obra

O acesso a 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:

1. Dados sobre a conferência, evento ou *journal* em que foi publicada a obra.
2. Número do DOI.
3. O abstract que apresenta um resumo da obra.
4. 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 pela *tag* "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.

```
int n = 1;
int itensPagina = 0;
String url = URLPaginaProfessor;
doc = Jsoup.connect(url);
int quantidade = doc.getElementsByClass("nova-o-stack__item pagination--top").first().text().toInteger();
#Acima o número de Research Items é transformado em inteiro

WHILE(itensPagina < quantidade)
    n++;
    itensPagina += 100;
    algoritmoComparacaoDeDados();

    IF(itensPagina < quantidade)
        doc = Jsoup.connect(url + "/" + n);
    END IF
END WHILE
```

Figura 21 - Pseudo-código do algoritmo de paginação do ResearchGate

4.2.3 ALGORITMO DE CONEXÃO COM OS SERVIDORES DE *PROXY*

Foi feito o *download* de forma manual de um arquivo contendo uma lista de IP´s de servidores brasileiros de *proxy* , esta lista foi filtrada para conter apenas servidores que possuem um valor de *uptime* maior ou igual que 50%, ou seja, servidores que estavam disponíveis na maioria das vezes que foi feita uma requisição.

A biblioteca Jsoup que é usada para fazer as requisições HTTP suporta o uso de *proxy* sendo necessário apenas informar o IP e porta do servidor que se deseja usar.

O *crawler* lê o arquivo e salva todos os IP´s em uma lista na memória, tenta acessar o repositório usando o IP do primeiro *proxy* encontrado na lista. Se o proxy estiver acessível incrementa uma variável que controla o número de requisições já feitas e faz a requisição, senão o IP é removido da lista e é pego o próximo IP na lista para tentar fazer a requisição, o mesmo acontece quando a variável que controla o número de requisições já feitas alcançar o valor de 80.

4.3 ESQUEMA DO BANCO DE DADOS

O banco de dados possui um papel fundamental neste trabalho pois os resultados de todo o processamento feito em cima dos dados é armazenado. Ele possui uma estrutura simples que permite o armazenamento das obras e também consultas em cima dos dados.

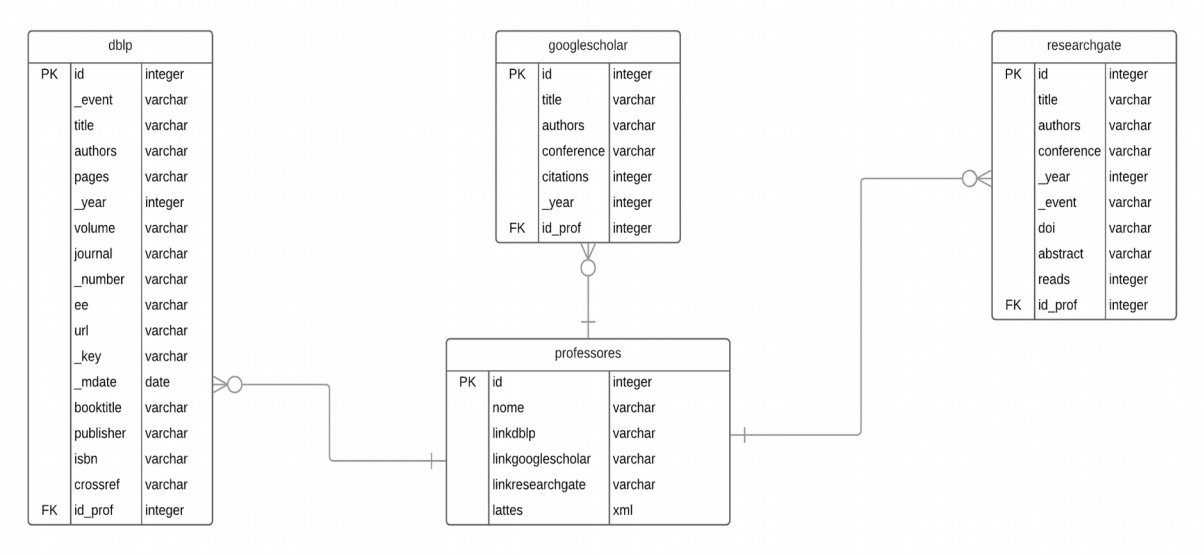


Figura 22 - Esquemático do banco de dados

O banco de dados possui quatro tabelas, a tabela googlescholar, researchgate e dblp armazenam dados das produções dos respectivos repositórios e referenciam a tabela professores através da FK(*Foreign Key*) “id_prof”.

A tabela professores contém todos os pesquisadores que o *crawler* irá comparar e extrair dados, ela é gerada através dos currículos lattes de cada pesquisador e armazena também o link para acessar a URL do pesquisador em cada um dos repositórios. Esses atributos foram criados com a intenção de o usuário armazenar manualmente a URL da página para garantir com certeza que a página com que o *crawler* irá trabalhar representa as obras do professor.

5 EXPERIMENTOS

Este capítulo tem o intuito de demonstrar como foram conduzidos os experimentos a fim de avaliar a qualidade do *crawler* em extrair novas obras que sejam relevantes para ampliar o currículo acadêmico dos pesquisadores.

É testada a capacidade do algoritmo utilizado nesta obra de persistir no banco de dados o maior número de obras relevantes possíveis tentando evitar cópias de obras que já estão contidas no lattes.

5.1 BASE DE DADOS

A base de dados conta com 58 professores do INE que possuem um currículo na plataforma do lattes, dos quais a grande maioria possui também dados das suas obras também registrados nas três fontes pesquisadas nesta obra.

O algoritmo leva aproximadamente 2 horas e 30 minutos para comparar as obras de todos os professores registrados no banco de dados, comparar com as existentes no lattes e então persistir o que for considerado relevante.

Foram persistidas 44 obras recuperadas da DBLP, 746 advindas do *GoogleScholar* e 340 advindas do *ResearchGate*, a menor base de dados de obras recuperadas é a da DBLP, porém apresenta ser a mais confiável visto que os dados das obras adicionados neste repositório são obtidos diretamente dos organizadores de conferências e eventos. O *GoogleScholar* por outro lado apresenta a maior quantidade de resultados porém com menor confiabilidade visto que várias obras são indexadas pelo *crawler* da *Google* conhecido como *Googlebot*.

5.2 VARIÁVEIS

Para avaliar o *crawler* são usadas variáveis que permitem identificar a relevância das obras recuperadas para o contexto de ampliar os currículos, as variáveis são as seguintes:

- **obrasDBLP**: Conjunto de obras recuperadas da DBLP.

- **obrasRelevantesDBLP**: Conjunto de obras recuperadas da DBLP que não estão contidas no currículo lattes do professor e que são obras que contém algum conteúdo novo.

- **obrasGoogleScholar**: Conjunto de obras recuperadas do *GoogleScholar*.

- **obrasRelevantsGoogleScholar**: Conjunto de obras recuperadas do *GoogleScholar* que não estão contidas no currículo lattes do professor.

- **obrasResearchGate**: Conjunto de obras recuperadas do ResearchGate.

- **obrasRelevantesResearchGate**: Conjunto de obras recuperadas do ResearchGate que não estão contidas no currículo lattes do professor.

- **totalObrasRelevantes**: A soma de todas as obras relevantes encontradas em todos os repositórios.

- **totalDeObrasLattes**: Conjunto total de todas as obras de todos os professores usadas para comparação pelo *crawler*.

5.3 MÉTRICAS

Foi utilizada uma métrica amplamente utilizada para fins de avaliar a recuperação de informação conhecida como precisão, que indica a porcentagem de documentos recuperados que são relevantes para o contexto desta obra.

Existem outras duas métricas que assim como a precisão também são muito utilizadas, como a revocação e a medida-f, porém elas não foram utilizadas nesta obra pelo fato de não serem significantes neste contexto, já que a revocação seria uma constante neste caso, já a medida-f é uma medida que depende da revocação e portanto também não acrescentaria em nada em termos de análise da eficiência do *crawler*.

Pode se concluir que uma medida mais interessante neste contexto seria

algo que indique o quão útil foi o *crawler* em ampliar os currículos, para isso foi usada uma medida que será chamada de recuperação, que indica o acréscimo em porcentagem do número de obras que foram adicionadas aos currículos.

$$\text{Precisão obrasDBLP} = \frac{|obrasRelevantesDBLP \cap obrasDBLP|}{|obrasDBLP|}$$

Equação 1 - Precisão das obras da DBLP

$$\text{Recuperação obrasDBLP} = \frac{100 \times obrasRelevantesDBLP}{totalDeObrasLattes}$$

Equação 2 - Recuperação das obras da DBLP

$$\text{Precisão obrasGoogleScholar} = \frac{|obrasRelevantesGoogleScholar \cap obrasGoogleScholar|}{|obrasGoogleScholar|}$$

Equação 3 - Precisão das obras do *GoogleScholar*

$$\text{Recuperação obrasGoogleScholar} = \frac{100 \times obrasRelevantesGoogleScholar}{totalDeObrasLattes}$$

Equação 4 - Recuperação das obras do *GoogleScholar*

$$\text{Precisão obrasRG} = \frac{|obrasRelevantesResearchGate \cap obrasResearchGate|}{|obrasResearchGate|}$$

Equação 5 - Precisão das obras do *ResearchGate*

$$\text{Recuperação obrasResearchGate} = \frac{100 \times obrasRelevantesResearchGate}{totalDeObrasLattes}$$

Equação 6 - Recuperação das obras da DBLP

$$\text{Recuperação obrasTotal} = \frac{100 \times totalObrasRelevantes}{totalDeObrasLattes}$$

Equação 7 - Recuperação do total de obras encontradas

5.4 RESULTADOS

Conforme mostra a tabela 2 a avaliação dos resultados contou com o conjunto completo de obras recuperadas da DBLP, 70 obras do *GoogleScholar* por ser o repositório do qual foram recuperadas um maior número de obras e por fim 50 obras do ResearchGate.

Repositório	**Quantidade de obras recuperadas usadas para avaliação**
DBLP	**44**
GoogleScholar	**70**
ResearchGate	**50**
Total	**164**

Tabela 2 - Quantidade de obras recuperadas usadas para realizar os experimentos

A precisão do *crawler* somando todos os repositórios atingiu 75.60%, já a recuperação do total de obras atingiu 18.18% que representa um acréscimo considerável no currículo dos professores visto que o lattes é mantido atualizado por muitos professores.

A dificuldade consiste na busca de uma melhor precisão possível, visto que o foco é tentar armazenar apenas obras consideradas relevantes.

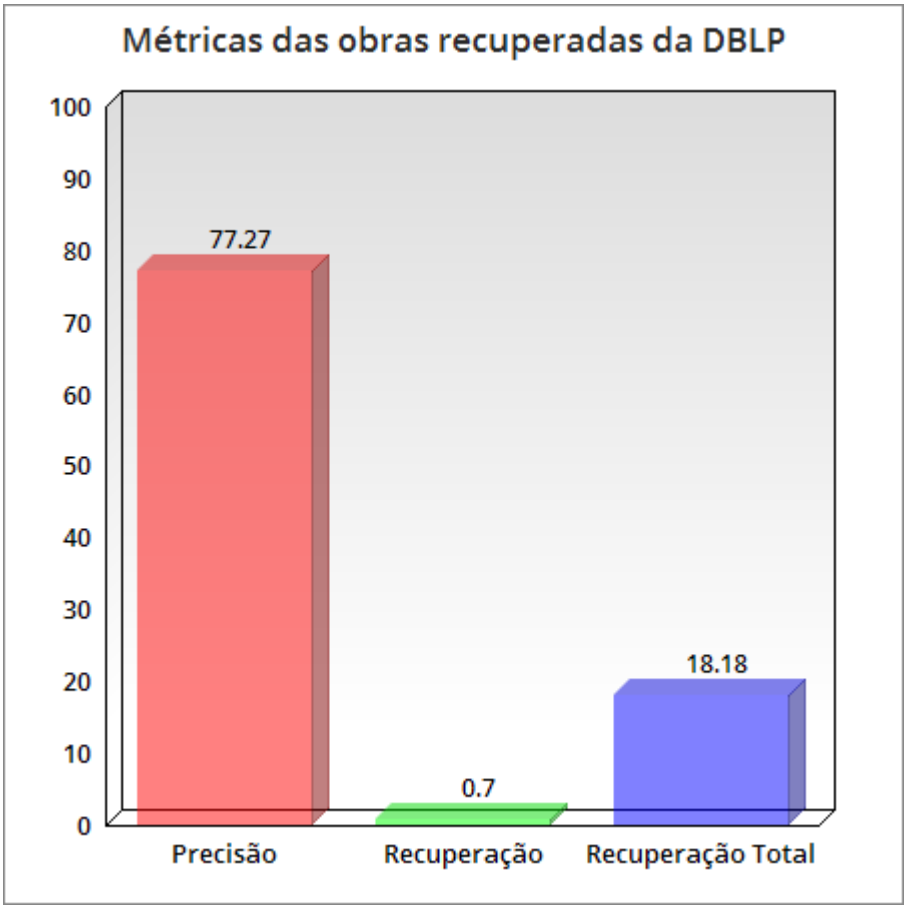


Figura 23 - Resultado dos testes realizados com as obras recuperadas da DBLP

A figura 23 revela os resultados para as obras recuperadas através do repositório da DBLP, dentro das 44 obras, 34 foram classificadas como relevantes no sentido de ampliar o lattes e as outras 10 obras foram duplicadas pois foram classificadas incorretamente como relevantes pelo algoritmo, assim a precisão foi de 77.27%.

A recuperação foi de apenas 0.7% que foi a menor de todas, porém apesar de este valor representar um pequeno acréscimo é importante notar que as obras recuperadas da DBLP, como citado anteriormente, possuem metadados que são inseridos por humanos, logo sua fonte é mais confiável do que o *GoogleScholar.*

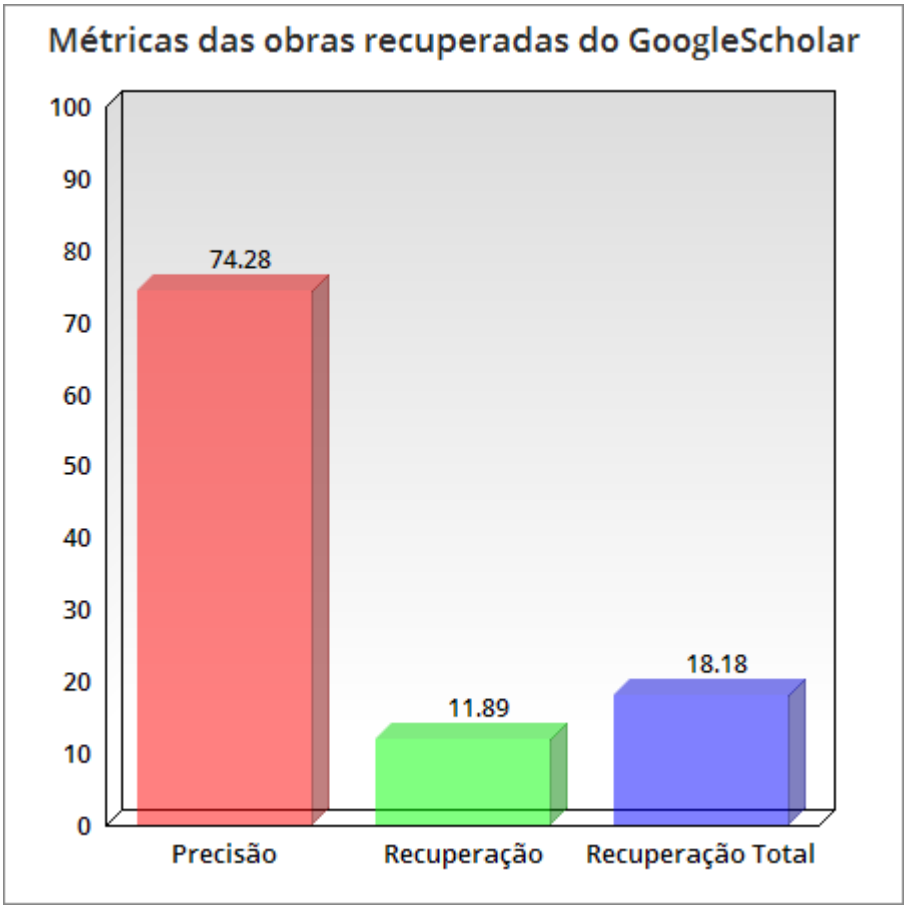


Figura 24 - Resultado dos testes realizados com as obras recuperadas do *GoogleScholar*

A figura 24 nos mostra os resultados das obras recuperadas do *GoogleScholar*, das 70 obras usadas para teste, 52 foram classificadas como relevantes e as outras 18 classificadas como réplicas, resultando assim em uma precisão de 74.28%.

É necessário notar as obras recuperadas do *GoogleScholar* são indexadas por um *crawler* e possuem menor credibilidade visto que algumas não possuem quase ou nenhum valor na relevância dos seus dados. Por exemplo, uma obra persistida no banco de dados possui o título de “Link to download the paper at IEEE Explore”, portanto é fácil notar que este registro não é uma obra mas foi indexado como tal pelo *GoogleScholar* e como resultado o ECL também identifica como uma obra relevante.

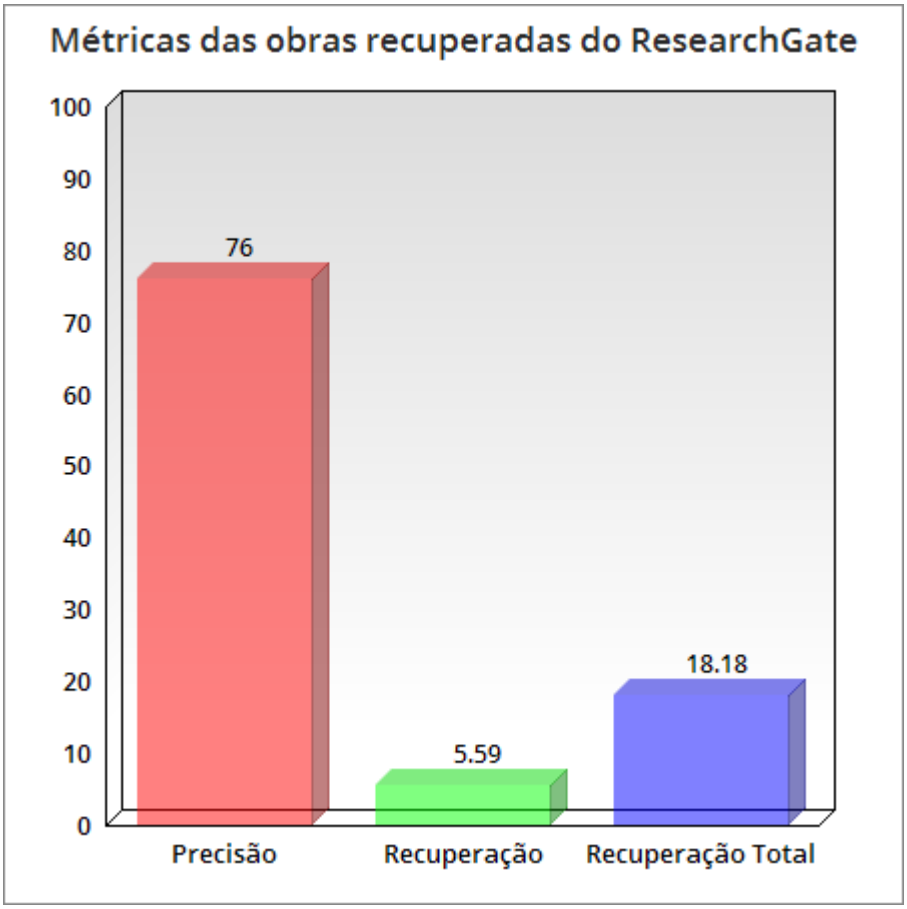


Figura 25 - Resultado dos testes realizados com as obras recuperadas do *ResearchGate*

A figura 25 nos mostra os resultados das obras recuperadas do *ResearhGate,* com uma precisão semelhante aos obtidos através das obras do *GoogleScholar*, das 50 obras usadas para teste, 38 foram classificadas como relevantes e as outras 12 classificadas como réplicas, resultando assim em uma precisão de 76%.

A maioria das obras relevantes encontradas podem ser classificadas como “Trabalhos Técnicos” que alguns professores não inserem em seus currículos lattes, também outras vinculadas como “Data” que contém resultados de experimentos realizados por um trabalho.

O *ResearchGate* também apresenta alguns metadados relevantes que não são encontrados sempre no lattes, como o “DOI” e o “Resumo” que contribuem para a completude inclusive das obras que foram classificadas como réplicas.

O algoritmo opera fazendo a comparação de um conjunto de caracteres que formam os metadados, sendo o mais importante deles o título, pelo fato de este

método conter falhas a precisão é consequentemente afetada por alguns fatores que são citados a seguir:

- Alguns títulos mesmo que tenham sido considerados bastante diferentes pelo algoritmo de comparar *Strings*, representavam semanticamente a mesma obra, assim o algoritmo decidiu persistir uma obra que considerava relevante.
- Mesmo quando os títulos são muito diferentes o algoritmo ainda compara o ano das obras e o conjunto de autores, pode acontecer de o ano estar discrepante e o nome dos autores estarem abreviados de maneiras diferentes, resultando no não reconhecimento de uma obra semelhante.
- Alguns títulos nos repositórios estavam escritos em inglês e no lattes em português ou vice-versa, assim foram identificados como sendo uma obra que não está contida no lattes e portanto foram persistidas.

É interessante notar também que mesmo que parte das obras tenham sido consideradas réplicas de outras obras do lattes alguns metadados como o número de páginas, volume ou conferência possuem divergências, indicando assim que a persistência desta obra no banco de dados não é de total irrelevância. Outro fator considerado importante é que existem alguns metadados que estão contidos nos repositórios pesquisados que não estão contidos no lattes, o que indica que toda obra que foi persistida no banco pode ter alguma informação relevante que visa ampliar o currículo lattes.

6 CONSIDERAÇÕES FINAIS

A Internet serve como uma biblioteca digital que armazena uma quantidade vasta de dados, entre eles dados referentes a publicações acadêmicas de pesquisadores. Assim foi criado o ECL com o intuito de ampliar e unificar os dados das obras contidas em diferentes repositórios em uma única aplicação.

Através da realização dos experimentos foi possível concluir que a ferramenta atingiu o seu objetivo. 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.

Ao longo do desenvolvimento do *crawler* notou-se os vários desafios inerentes a criação de uma ferramenta que coleta, processa e armazena dados de forma automatizada. A coleta de dados representa um desafio em relação ao estudo das páginas que serão analisadas e ao tratamento adequado que é necessário se fazer em cima dos dados para evitar erros na ferramenta. As páginas mudam constantemente e por isso é necessário um cuidado especial em fazer a extração correta dos dados.

Também hoje em dia com o crescimento do interesse em coletar dados de forma automatizada, houve o crescimento do números de *crawlers* vasculhando dados pela Internet e por isso vários sites impõem limites de acesso para evitar a sobrecarga dos seus servidores causadas pelos *crawlers.* Assim foi necessária uma estratégia para tratar esses casos.

O algoritmo de comparação de dados também se mostra muito importante visto que o ideal seria coletar o máximo possível de dados novos sem redundância, tarefa que se mostra complexa já que os dados são estruturados de forma diferentes através dos vários repositórios.

O banco de dados também necessita de atenção pois precisa ser modelado corretamente e os dados persistidos precisam ser tratados de acordo com os seus tipos. A aplicação precisa ser tão intuitiva quanto puder já que a usabilidade também representa um fator importante na avaliação de uma aplicação.

6.1 TRABALHOS FUTUROS

A criação de um *crawler* atinge um amplo espectro de áreas do conhecimento. Por questões de tempo e redução da complexidade do escopo do trabalho algumas ideias não foram desenvolvidas ou possivelmente foram implementadas de maneira sub-ótima. A seguir são listadas sugestões de melhorias e ideias que visam melhorar a qualidade da aplicação.

- **Melhoria no algoritmo de comparação de dados**: Por simplicidade, o método escolhido para comparar as obras foi o uso da comparação de strings, esta alternativa está muito longe de apresentar um resultado ótimo. Assim como visto anteriormente o *crawler* do Cite Seer X foi testado usando técnicas de *machine learning* como SVM, Naive Bayes e outros modelos que se apresentam muito mais adequados a resolução deste problema. Redes neurais também têm se apresentado extremamente úteis para resolver diversos tipos de problemas, em especial, as redes neurais convolucionais que vêm sendo grande alvo de estudos e aplicações nos últimos anos.
- **Refinamento na seleção dos dados extraídos**: Este *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.
- **Otimização 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 Tor *Browser* ou de outras maneiras não pensadas, pode ser possível reduzir o tempo de busca em dezenas ou até centenas de vezes.
- **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.

- **Informar os professores sobre as obras faltantes**: Assim como é feito no SiSoB, poderia ser criado um módulo que extrai e envia e-mails aos professores informando-os das obras recuperadas e ajudando a manter o currículo atualizado.
- **Ampliar o alcance das obras comparadas do lattes**: O currículo lattes é muito vasto e detalhado e apresenta um número muito grande de possíveis seções que contém diferentes dados. Neste trabalho foram investigadas as seções contendo: artigos, trabalhos em eventos, livros, capítulos de livros e outras produções. Apesar de estas seções conterem a grande maioria de dados relevantes é possível expandir o alcance de comparações de obras do lattes para evitar réplicas salvas no banco de dados.
- **Expansão do escopo de busca**: Acrescentar novos repositórios irá aumentar o número de obras recuperadas e contribuir com a atualização dos currículos investigados.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] CNPQ. Sobre a Plataforma. Disponível em <http://lattes.cnpq.br/web/plataforma-lattes/home>. Acesso em: 24 de novembro de 2018.

[2] REPANOVICI, Angela. Measuring the visibility of the university's scientific production through scientometric methods: An exploratory study at the Transilvania University of Brasov, Romania. Performance Measurement and Metrics. v.12, n.2, pp.106-117, 2011.

[3] PINKERTON, Brian. WebCrawler: Finding What People Want, Dissertação (Doutorado) - University of Washington, USA, 2000.

[4] DIKAIAKOS, Marios D; STASSOPOULOU, Athena; PAPAGEORGIOU, Loizos. An investigation of web crawler behavior: characterization and metrics. Computer Communications. v.28, n.8, pp.880-897, 2005.

[5] CHAKRABARTIA, Soumen; BERG, Martin van den; DOM, Byron. Focused crawling: a new approach to topic-specific Web resource discovery. Computer Networks. v.31, n.11-16, pp.1623-1640, 1999.

[6] Wikipedia, Dados, Disponível em <https://pt.wikipedia.org/wiki/Dados> Acesso em: 24 de novembro de 2018.

[7] CRESCENZI, Valter; MECCA, Giansalvatore; MERIALDO, Paolo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. *VLDB*, 2001.

[8] GEUNA, Aldo et al. SiSOB data extraction and codification: A tool to analyze scientific careers. Research Policy. v.44, n.9, pp.1645-1658, 2015.

[9] LUOTONEN, Ari; ALTIS, Kevin. World-Wide Web Proxies. Computer Network and ISDN Systems. v.22, n.2, pp.147-154, 1994.

[10] BISHOP, Christopher. Pattern Recognition and Machine Learning. Information Science and Statistics. ISSN: 1613-9011, 2006.

[11] SHKAPENYUK, Vladislav; SUEL, Torsten. Design and Implementation of a High-Performance Distributed Web Crawler. In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, pp.357-368 2002.

[12] WU, Jian et al. CiteSeerX: AI in a Digital Library Search Engine. AI Magazine. v.36, pp.35-48, 2014.

[13] RULLO, Michele. A Flexible Architecture for Secure and Anonymous Web Crawling, Dissertação (Mestrado), Sapienza Università di Roma, Itália, 2016.

APÊNDICE A - CONFIGURAÇÃO DO BANCO DE DADOS USADO NOS EXPERIMENTOS

O banco de dados usado para realizar os experimentos precisa ser populado previamente através da tabela Professores para que o *crawler* consiga acessar os *links* que contém as suas obras nos três repositórios. Foram inseridos 58 professores do INE que continham um currículo lattes e eles foram inseridos usando os seguintes comandos SQL:

```
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (1, 'Aldo von Wangenheim', 'https://dblp.uni-trier.de/pers/xx/w/Wangenheim:Aldo_von.xml', 'https://scholar.google.com.br/citations?user=a7XTMeIAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Aldo_Von_Wangenheim2');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (2, 'Antonio Augusto Medeiros Fröhlich', 'https://dblp.uni-trier.de/pers/xx/f/Fr=ouml=hlich:Ant=ocirc=nio_Augusto.xml', 'https://scholar.google.com.br/citations?user=_oy2EqkAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Antonio_Froehlich');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (3, 'Antonio Carlos Mariani', 'https://dblp.uni-trier.de/pers/xx/m/Mariani:Antonio_Carlos.xml', null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (4, 'Carina Friedrich Dorneles', 'https://dblp.uni-trier.de/pers/xx/d/Dorneles:Carina_F=.xml', 'https://scholar.google.com.br/citations?user=yERABKUAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Carina_Dorneles');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (5, 'Christiane Anneliese Gresse von Wangenheim', 'https://dblp.uni-trier.de/pers/xx/w/Wangenheim:Christiane_Gresse_von.xml', 'https://scholar.google.com.br/citations?user=SXkE1UkAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Christiane_Gresse_von_Wangenheim');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
```

```
VALUES (6, 'Frank Augusto Siqueira',
'https://dblp.uni-trier.de/pers/xx/s/Siqueira:Frank.xml',
'https://scholar.google.com.br/citations?user=AlUvmiYAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Frank_Siqueira');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (7, 'José Luís Almada Güntzel',
'https://dblp.uni-trier.de/pers/xx/g/G=uuml=ntzel:Jos=eacute=_Lu=iacute=s_Almada.
xml', 'https://scholar.google.com.br/citations?user=0t1z_OkAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Jose_Luis_Guentzel');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (8, 'Jean Carlo Rossa Hauck',
'https://dblp.uni-trier.de/pers/xx/h/Hauck:Jean_C=_R=.xml',
'https://scholar.google.com.br/citations?user=fT5ItcEAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Jean_Hauck');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (9, 'Jean Everson Martina',
'https://dblp.uni-trier.de/pers/xx/m/Martina:Jean_Everson.xml',
'https://scholar.google.com.br/citations?user=Ov5Fe5QAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Jean_Martina');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (10, 'Luciana de Oliveira Rech',
'https://dblp.uni-trier.de/pers/xx/r/Rech:Luciana.xml',
'https://scholar.google.com.br/citations?user=1dMje98AAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Luciana_Rech');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (11, 'Olinto Jose Varela Furtado',
'https://dblp.uni-trier.de/pers/xx/f/Furtado:Olinto_J=_V=.xml',
'https://scholar.google.com.br/citations?user=14ZW_z8AAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Olinto_Furtado');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (12, 'Raul Sidnei Wazlawick',
'https://dblp.uni-trier.de/pers/xx/w/Wazlawick:Raul_Sidnei.xml',
'https://scholar.google.com.br/citations?user=zbaev20AAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Raul_Wazlawick');
```

```
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (13, 'Ricardo Azambuja Silveira',
'https://dblp.uni-trier.de/pers/xx/s/Silveira:Ricardo_Azambuja.xml',
'https://scholar.google.com.br/citations?user=hSsOFmIAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Ricardo_Silveira3');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (14, 'Ricardo Felipe Custodio',
'https://dblp.uni-trier.de/pers/xx/c/Cust=oacute=dio:Ricardo_Felipe.xml',
'https://scholar.google.com.br/citations?user=l29lg_IAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Ricardo_Custodio');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (15, 'Roberto Willrich',
'https://dblp.uni-trier.de/pers/xx/w/Willrich:Roberto.xml',
'https://scholar.google.com.br/citations?user=klG5GT4AAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Roberto_Willrich');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (16, 'Silvia Modesto Nassar',
'https://dblp.uni-trier.de/pers/xx/n/Nassar:Silvia_M=.xml', null,
'https://www.researchgate.net/profile/Silvia_Nassar2');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (17, 'Carlos Becker Westphall',
'https://dblp.uni-trier.de/pers/xx/w/Westphall:Carlos_Becker.xml','https://
scholar.google.com.br/citations?user=ah0TWZIAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Carlos_Westphall');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (18, 'Adriano Ferreti Borgatto',
'https://dblp.uni-trier.de/pers/xx/b/Borgatto:Adriano_Ferreti.xml', null,
'https://www.researchgate.net/profile/Adriano_Borgatto');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (19, 'Alexandre Gonçalves Silva',
'https://dblp.uni-trier.de/pers/xx/s/Silva:Alexandre_Gon=ccedil=alves.xml','https://
scholar.google.com.br/citations?user=ByzwM6wAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Alexandre_Silva10');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
```

```
VALUES (20, 'Alex Sandro Roschildt Pinto',
'https://dblp.uni-trier.de/pers/xx/p/Pinto:Alex_R=.xml',
'https://scholar.google.com.br/citations?user=bQhXm9sAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Alex_Pinto3');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (21, 'Andrea Cristina Konrath',
'https://dblp.uni-trier.de/pers/xx/k/Konrath:Andrea_Cristina.xml',null , null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (22, 'Andreia Zanella',
'https://dblp.uni-trier.de/pers/xx/z/Zanella:Andreia.xml', null,
'https://www.researchgate.net/profile/Andreia_Zanella');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (23, 'André Wüst Zibetti', null,'https://scholar.google.com.br/citations?
user=H7UpYfUAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Andre_Wuest_Zibetti');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (24, 'Arthur Ronald de Vallauris Buchsbaum',
'https://dblp.uni-trier.de/pers/xx/b/Buchsbaum:Arthur.xml',
'https://scholar.google.com.br/citations?user=-8xNJ6UAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Arthur_Buchsbaum');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (25, 'Carla Merkle Westphall',
'https://dblp.uni-trier.de/pers/xx/w/Westphall:Carla_Merkle.xml','https://
scholar.google.com.br/citations?user=L74FEpQAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Carla_Merkle_Westphall');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (26, 'Cristina Meinhardt',
'https://dblp.uni-trier.de/pers/xx/m/Meinhardt:Cristina.xml',
'https://scholar.google.com.br/citations?user=nSxMVPkAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Cristina_Meinhardt');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (27, 'Daniel Santana de Freitas',
'https://dblp.uni-trier.de/pers/xx/f/Freitas:Daniel_Santana_de.xml', null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
```

```
VALUES (28, 'Elder Rizzon Santos',
'https://dblp.uni-trier.de/pers/xx/s/Santos:Elder_Rizzon.xml', null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (29, 'Fabiane Barreto Vavassori Benitti',
'https://dblp.uni-trier.de/pers/xx/b/Benitti:Fabiane_Barreto_Vavassori.xml','https://
scholar.google.com.br/citations?user=c2CrjZkAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Fabiane_Benitti');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (30, 'Fernando Augusto da Silva Cruz',
'https://dblp.uni-trier.de/pers/xx/c/Cruz:Fernando_Augosto_da_Silva.xml',
'https://scholar.google.com.br/citations?user=oJEFHukAAAAJ&hl=pt-BR', null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (31, 'Jerusa Marchi',
'https://dblp.uni-trier.de/pers/xx/m/Marchi:Jerusa.xml','https://scholar.google.com.br/
citations?user=tGw15cMAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Jerusa_Marchi');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (32, 'João Bosco Mangueira Sobral',
'https://dblp.uni-trier.de/pers/xx/s/Sobral:Jo=atilde=o_Bosco_M=.xml',
'https://scholar.google.com.br/citations?user=dJqONLoAAAAJ&hl=pt-BR',
'https://www.researchgate.net/profile/Joao_Bosco_Sobral');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (33, 'Joao Candido Lima Dovicchi',
'https://dblp.uni-trier.de/pers/xx/d/Dovicchi:Joao.xml', null,
'https://www.researchgate.net/profile/Joao_Dovicchi');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (34, 'Jose Eduardo De Lucca', null, null,
'https://www.researchgate.net/profile/Jose_Eduardo_De_Lucca');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (35, 'Juliana Eyng', 'https://dblp.uni-trier.de/pers/xx/e/Eyng:Juliana.xml',
null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate)
VALUES (36, 'Laércio Lima Pilla',
'https://dblp.uni-trier.de/pers/xx/p/Pilla:La=eacute=rcio_Lima.xml',
```

'https://scholar.google.com.br/citations?user=i189aV0AAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Laercio_Lima_Pilla');

INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (37, 'Leandro José Komosinski', null, null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (38, 'Lúcia Helena Martins-Pacheco', 'https://dblp.uni-trier.de/pers/xx/p/Pacheco:L=uacute=cia_Helena_Martins.xml', null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (39, 'Luiz Cláudio Villar dos Santos', 'https://dblp.uni-trier.de/pers/xx/s/Santos:Luiz_Cl=aacute=udio_Villar_dos.xml', null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (40, 'Luiz Ricardo Nakamura', null, null, 'https://www.researchgate.net/profile/Luiz_Nakamura2');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (41, 'Maicon Rafael Zatelli', 'https://dblp.uni-trier.de/pers/xx/z/Zatelli:Maicon_Rafael.xml', null, 'https://www.researchgate.net/profile/Maicon_Zatelli');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (42, 'Marcelo Menezes Reis', 'https://dblp.uni-trier.de/pers/xx/r/Reis:Marcelo_Menezes.xml', null, 'https://www.researchgate.net/profile/Marcelo_Reis7');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (43, 'Márcio Bastos Castro', 'https://dblp.uni-trier.de/pers/xx/c/Castro:M=aacute=rcio_Bastos.xml', 'https://scholar.google.com.br/citations?user=3sxkal4AAAAJ&hl=pt-BR', null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (44, 'Maurício Floriano Galimberti', null, null, 'https://www.researchgate.net/profile/Mauricio_Floriano_Galimberti');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (45, 'Mauro Roisenberg', 'https://dblp.uni-trier.de/pers/xx/r/Roisenberg:Mauro.xml',

'https://scholar.google.com.br/citations?user=qmyvtaYAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Mauro_Roisenberg');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (46, 'Meire Mezzomo', null, null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (47, 'Patricia Della Méa Plentz', 'https://dblp.uni-trier.de/pers/xx/p/Plentz:Patricia_Della_Mea.xml', 'https://scholar.google.com.br/citations?user=mNmSmvUAAAAJ&hl=pt-BR', null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (48, 'Patrícia Vilain', 'https://dblp.uni-trier.de/pers/xx/v/Vilain:Patricia.xml', null, 'https://www.researchgate.net/profile/Patricia_Vilain');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (49, 'Pedro Alberto Barbetta', 'https://dblp.uni-trier.de/pers/xx/b/Barbetta:Pedro_Alberto.xml', 'https://scholar.google.com.br/citations?user=b7ETU6IAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Pedro_Barbetta');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (50, 'Rafael Luiz Cancian', 'https://dblp.uni-trier.de/pers/xx/c/Cancian:Rafael_Luiz.xml', null, 'https://www.researchgate.net/profile/Rafael_Cancian');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (51, 'Renato Cislaghi', 'https://dblp.uni-trier.de/pers/xx/c/Cislaghi:Renato.xml', 'https://scholar.google.com.br/citations?user=uN3n8tMAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Renato_Cislaghi');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (52, 'Renato Fileto', 'https://dblp.uni-trier.de/pers/xx/f/Fileto:Renato.xml', 'https://scholar.google.com.br/citations?user=kO9RRxgAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Renato_Fileto2');
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (53, 'Ricardo Pereira e Silva', 'https://dblp.uni-trier.de/pers/xx/s/Silva:Ricardo_Pereira_e.xml', null, null);
INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (54, 'Ronaldo dos Santos Mello',

'https://dblp.uni-trier.de/pers/xx/m/Mello:Ronaldo_dos_Santos.xml', null, null);

INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (55, 'Sergio Peters', null, null, 'https://www.researchgate.net/profile/Sergio_Peters');

INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (56, 'Vania Bogorny', 'https://dblp.uni-trier.de/pers/xx/b/Bogorny:Vania.xml', 'https://scholar.google.com.br/citations?user=-x_FW-wAAAAJ&hl=pt-BR', 'https://www.researchgate.net/profile/Vania_Bogorny');

INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (57, 'Vera do Carmo Comparsi de Vargas', 'https://dblp.uni-trier.de/pers/xx/v/Vargas:Vera_do_Carmo_C=_de.xml', null, null);

INSERT into professores (id, nome, linkdblp, linkgooglescholar, linkresearchgate) VALUES (58, 'Vitório Bruno Mazzola', null, null, null);

APÊNDICE B - CÓDIGO FONTE

O código-fonte foi hospedado no GitHub e pode ser acessado através do seguinte *link*: “https://github.com/arthurmbranco/CrawlerTCC”

APÊNDICE C – ARTIGO SBC

FERRAMENTA PARA COLETA E COMPARAÇÃO DE DADOS DE PUBLICAÇÕES ACADÊMICAS COM O CURRÍCULO LATTES

Arthur Machado Branco[1]
[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC - Brazil

arthur.branco93@gmail.com

Abstract. *As a way to evaluate professional skills and the amount of academic work done by a researcher the lattes curricula was created by CNPq and is regarded as a national standard to store information about academic career and therefore is widely adopted by national academic institutions [1]. This work proproses the development of a tool that extracts relevant information of external data sources to complement the lattes curricula and combine data from both sources to help the user´s view and comprehension about the data retrieved.*

Resumo. *Como uma forma de avaliar as competências e a quantidade de obras produzidas por pesquisadores o currículo lattes foi criado pelo CNPq e é considerado como um padrão nacional para o registro da vida acadêmica e portanto é adotado por um grande número de instituições no país [1]. Este trabalho tem como objetivo a criação de uma ferramenta que faça a extração de informações relevantes de outras fontes de dados a fim de complementar o currículo lattes e que integre e unifique os dados de ambas as fontes para facilitar a visão e compreensão do usuário acerca dos dados que foram recuperados.*

1. INTRODUÇÃO

Existem diversas métricas para avaliar um pesquisador dentro do mundo acadêmico como: número total de publicações, número de citações, *H-index*, *G-index*, *HC-index* e *HI-norm* [2]. Para que se aplique as métricas descritas de forma ideal é necessário conter o conjunto completo de obras produzidas por um pesquisador.

Atualmente o currículo lattes é um padrão nacional usado por grande parte dos pesquisadores para armazenar dados referentes a todo seu histórico de vida acadêmico [1]. Estes dados são adicionados de forma manual ao currículo lattes e podem não refletir todo o potencial do pesquisador pois não é comum a eles manterem o currículo totalmente atualizado constantemente.

O currículo lattes é de grande importância e por ser bastante requisitado por instituições pode ser a porta de entrada para participação do pesquisador em grandes projetos de pesquisa, formação de grupos de trabalhos, participação em eventos e até mesmo oportunidade de emprego. É essencial que esteja completo já que a falta de informações pode ser a diferença entre alcançar ou não um objetivo almejado.

A internet disponibiliza a todos o acesso a grandes quantidades de dados de vários tipos, incluindo vários sites especializados em armazenar dados sobre as produções dos pesquisadores. Para trabalhar com tantos dados de forma eficaz precisamos usar ferramentas de software especializadas em coletar dados da web, que são conhecidas como *webcrawlers* [3].

Apesar de o auxílio de um *webcrawler* ser essencial neste tipo de tarefa, seu uso tem que ser bem planejado, se a pesquisa pelos dados for feita de forma mais abrangente pesquisando através de motores de busca que indexam quantidades gigantes de dados a qualidade destas buscas será pior. Quando o domínio é muito extenso as buscas por dados tem que ser feitas de maneira mais genérica possível, por isso este trabalho se propõe a criar um *crawler* focado em alguns sites específicos que contém apenas os dados de interesse, que são as obras dos pesquisadores.

A coleta de dados será feita a partir de três sites com destaque no mundo acadêmico que são: DBLP, GoogleScholar e ResearchGate. É possível comparar seus dados com os dados de publicações do currículo lattes a fim de criar uma visão unificada que tem como objetivo armazenar a maior quantidade possível de produções dos pesquisadores para que suas competências possam ser avaliadas por completo.

2. TRABALHOS RELACIONADOS

O SiSOB [4] é uma ferramenta que se propõe a coletar, extrair e persistir informações advindas de diversas fontes da web sobre pesquisadores e agrupar tudo em um banco de dados que permita uma visão unificada dos dados de todos os pesquisadores, forneça análises estatísticas e também permite a manipulação dos dados para que os interessados possam usá-los para outros fins.

A ferramenta é dividida em cinco módulos que ao final da execução do último módulo gera um arquivo contendo um currículo que possui informações sobre dados pessoais, informações sobre as universidades de afiliação, campos de pesquisa, informações profissionais e dados sobre publicações.

O Cite Seer X [5] é uma biblioteca digital que possui um motor de busca que permite o usuário ter acesso a milhões de arquivos sobre as publicações de pesquisadores. Tem seu foco principal na literatura relacionada à computação e ciência da informação e permite ao usuário ter acesso a metadados como: título, autores, abstract e outros dados relacionados a publicações e também fazer o download em PDF das publicações.

O *crawler* desenvolvido para o Cite Seer X chamado citeseerxbot é um *crawler* focado e vasculha a web em busca de novos dados para serem indexados, a busca é feita

através de um arquivo contendo URL´s de domínios acadêmicos que são adicionadas diariamente de forma incremental.

3. ECL

Neste capítulo será detalhado o funcionamento da ferramenta ECL (Extensor do Currículo Lattes) que irá acessar as publicações de três repositórios diferentes e comparar seus dados com as publicações do currículo lattes. Ao longo do capítulo será mostrada a arquitetura do sistema e os detalhes de implementação de todos os módulos envolvidos.

3.1 VISÃO GERAL

Como pode ser visto na figura 1 a arquitetura da ferramenta pode ser divida em quatro etapas:
1 - Conexão com um servidor de *proxy* que irá intermediar as requisições enviadas aos repositórios do GoogleScholar e ResearchGate.

2 - Acesso aos repositórios da DBLP, GoogleScholar e ResearchGate e extração dos dados das publicações.

3 - Análise sintática do currículo lattes para comparação com os dados extraídos dos repositórios.

4 - Persistência no banco de dados do currículo lattes usado para comparação e das publicações que o *crawler* julgar que não estão contidas no lattes.

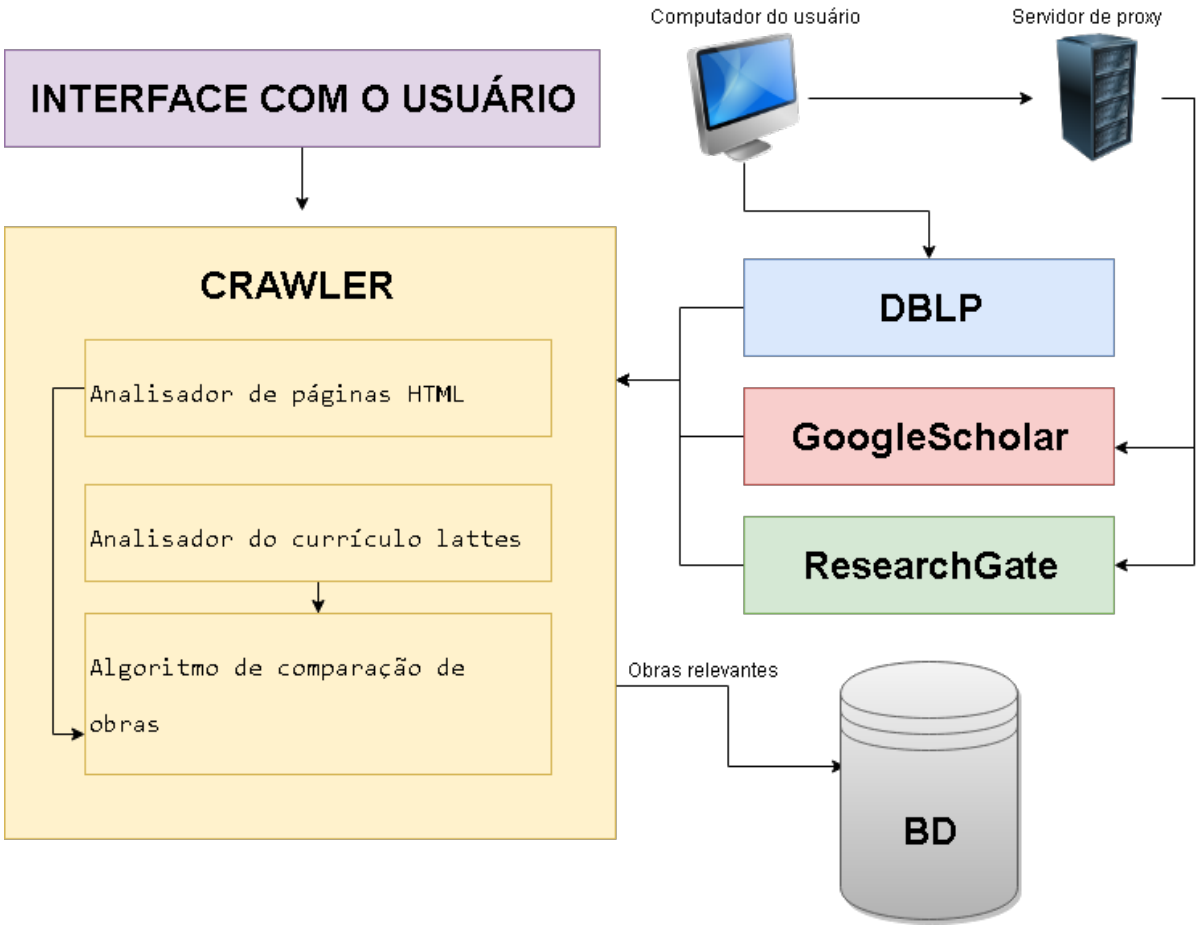


Figura 1. Arquitetura do ECL

Após o fim do processo de extração e coleta de dados o usuário pode através da interface, visualizar os dados recuperados e os dados contidos no currículo lattes de cada professor.

3.2. IMPLEMENTAÇÃO

A figura 2 mostra a interface de usuário da ferramenta, através de *checkboxes* presentes no canto superior esquerdo é possível selecionar os repositórios dos quais se deseja extrair informação. O botão de Pesquisar inicia a pesquisa que será feita a partir dos repositórios escolhidos.

No canto direito é apresentada uma lista de pesquisadores dos quais a ferramenta analisou o currículo lattes, selecionando um nome da lista a ferramenta carrega dados das obras pertencentes ao currículo lattes e também das obras que foram recuperadas de fontes externas, caso a pesquisa já tenha sido feita pelo menos uma vez.

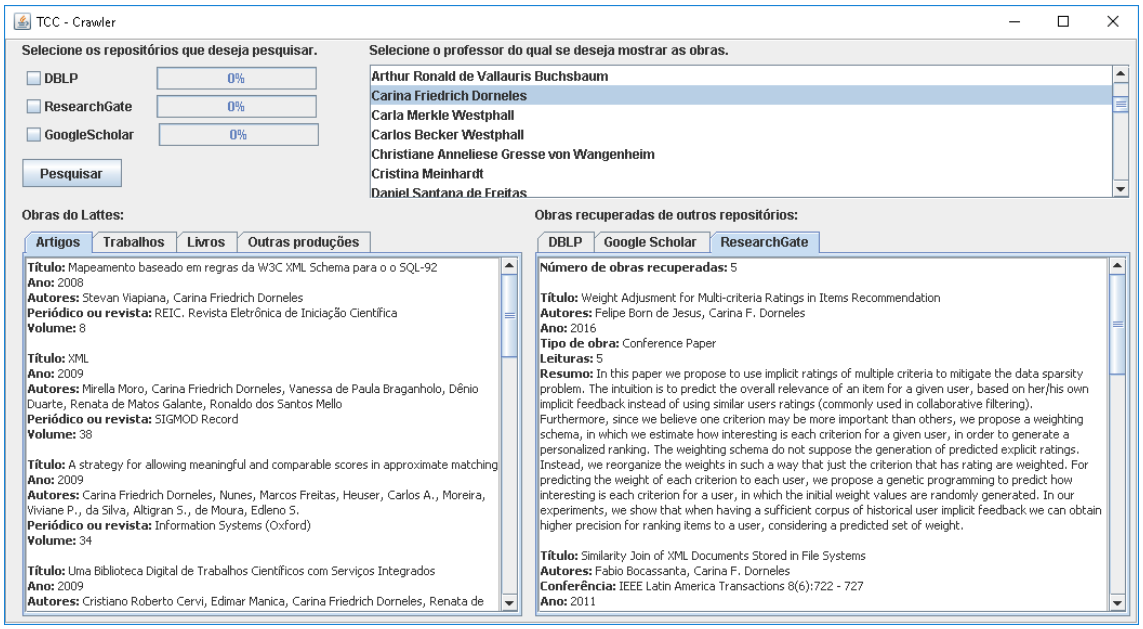


Figura 2. Interface gráfica do ECL

3.2.1. CONEXÃO COM SERVIDORES DE PROXY

Devido ao fato de o acesso aos repositórios do GoogleScholar e ResearchGate serem limitados, impedindo que várias requisições sejam feitas em um certo espaço de tempo, foi criado um algoritmo que faz a requisição a esses servidores através do uso de um servidor de proxy.

A ferramenta possui uma lista de servidores de proxy e usa o primeiro servidor da lista para fazer o acesso aos repositórios, se o servidor estiver indisponível o elimina da lista e tenta acessar o próximo. Se o próximo servidor estiver disponível faz o acesso aos repositórios através deste servidor escolhido até ele ficar indisponível ou até que sejam feitas 80 requisições através deste servidor.

3.2.2. EXTRAÇÃO E ANÁLISE DE DADOS

É necessário que o *download* dos currículos lattes dos pesquisadores seja feito manualmente no formato XML, assim a ferramenta pode fazer a análise sintática do arquivo que está estruturado em formato de árvore e possui vários metadados que indicam o caminho para acessar os dados relevantes sobre as obras dos pesquisador.

A extração dos dados das páginas funciona do mesmo modo no caso do GoogleScholar e ResearchGate em que o acesso aos dados é feito através de tags HTML, os dados das obras são estruturados em forma de árvore e estudando cada página individualmente é possível saber qual dado está contido dentro de cada *tag*.

Esta abordagem que visa extrair os dados para a memória, através da análise de *tags* contidas em uma página HTML, é mais fácil de desenvolver por não necessitar de nenhuma técnica avançada como o uso de algoritmos de IA para identificar os metadados nas páginas. Tem a desvantagem de ser inflexível pois se a estrutura das *tags* for mudada na página ou no

arquivo o *crawler* não consegue identificar onde se encontram os elementos e será necessário uma intervenção manual para descobrir novamente como funciona a estrutura da árvore DOM analisada.

Para a DBLP não é necessário acessar as *tags* da página em HTML pois este repositório permite o acesso dos dados estruturados em XML. Assim como o currículo lattes, pela facilidade de trabalhar com o XML é feito o *download* dos dados em XML para posteriormente serem analisados.

3.2.3. COMPARAÇÃO E PERSISTÊNCIA DE DADOS

Com os dados estruturados na memória é feito uma comparação de todos com todos para identificar quais dados são diferentes e devem ser persistidos, por exemplo, iterar sobre todas as obras da DBLP verificando se a obra é semelhante a alguma obra do currículo lattes iterando por todas as obras do lattes.

A ferramenta possui interesse em persistir apenas novas obras que não estejam contidas no currículo lattes analisado e para isso foi criado um algoritmo com a finalidade de comparar duas obras, que informa se elas são ou não semelhantes.

O algoritmo usado para comparar os metadados entre as fontes foi criado de forma empírica e tem como entrada metadados semelhantes das duas fontes comparadas como: título, ano e autores.

A comparação é feita através do uso de algoritmo de similaridade entre *strings*, foi usado o algoritmo LCS (*Longest Common Subsequence*) que consiste em achar a maior subsequência comum entre duas ou mais sequências, tem como entrada duas *strings* e sua saída é um valor normalizado entre 0 e 1, sendo que 0 representa duas *strings* iguais e 1 totalmente diferentes.

Como pode ser visto no fluxograma da figura 3 o algoritmo retorna TRUE caso as obras sejam semelhantes e FALSE caso elas sejam diferentes, se o retorno do algoritmo for FALSE então a obra do repositório que está sendo analisada é persistida no banco de dados pois representa uma informação recuperada que irá complementar o currículo lattes em questão.

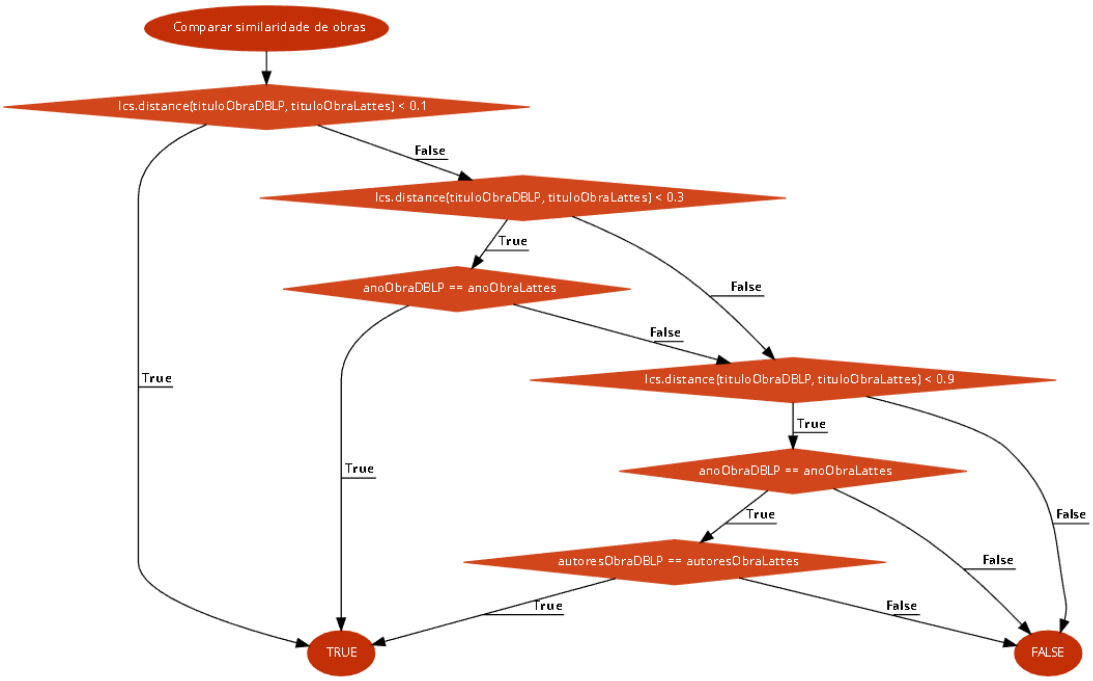


Figura 3. Algoritmo que compara a similaridade de duas obras

4. EXPERIMENTOS

A base de dados conta com 58 pesquisadores do Departamento de Informática e Estatística (INE/UFSC) que possuem um currículo na plataforma do lattes, dos quais a grande maioria possui dados das suas obras também registrados nas três fontes pesquisadas nesta obra.

Para medir o desempenho da ferramenta foi utilizada uma métrica muito conhecida chamada de precisão, que indica a porcentagem de documentos recuperados que são relevantes para o contexto desta obra.

Para o contexto desta obra em específico foi desenvolvida uma métrica que será chamada de índice de recuperação que pode ser visualizada na figura 4.

Essa métrica permite avaliar o quão útil é a função da ferramenta em incrementar os currículos lattes analisados. Indica o acréscimo em porcentagem do número de obras que foram adicionadas aos currículos, obras relevantes são as obras recuperadas dos repositórios que não são réplicas de obras contidas no lattes. A variável que está no denominador da equação da figura 4 representa o número total de obras que a ferramenta avaliou e submeteu a comparação de obras.

$$\text{Índice de recuperação} = \frac{100 \times obrasRelevantes}{totalDeObrasLattes}$$

Figura 4. Índice de recuperação

Conforme a tabela 1 foram avaliadas uma parcela do total de obras recuperadas que representam um total de 164 obras.

Tabela 1. Quantidade de obras recuperadas utilizadas no experimento

Repositório	Quantidade de obras recuperadas usadas para avaliação
DBLP	**44**
GoogleScholar	**70**
ResearchGate	**50**
Total	**164**

A precisão total do *crawler* atingiu 75.60%, já o índice de recuperação total de obras atingiu 18.18%, o que representa um acréscimo considerável no currículo dos pesquisadores. Na figura 5, 6 e 7 podem ser vistos os índices de recuperação individuais assim como a precisão atingida em cada repositório.

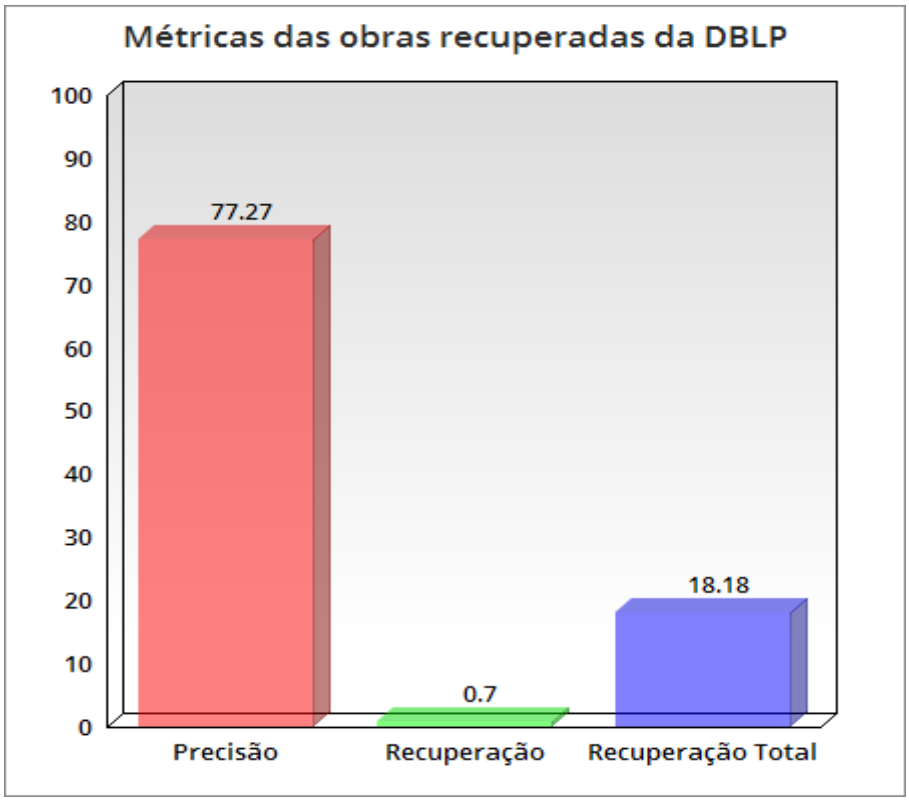


Figura 5. Resultado dos testes realizados com as obras da DBLP

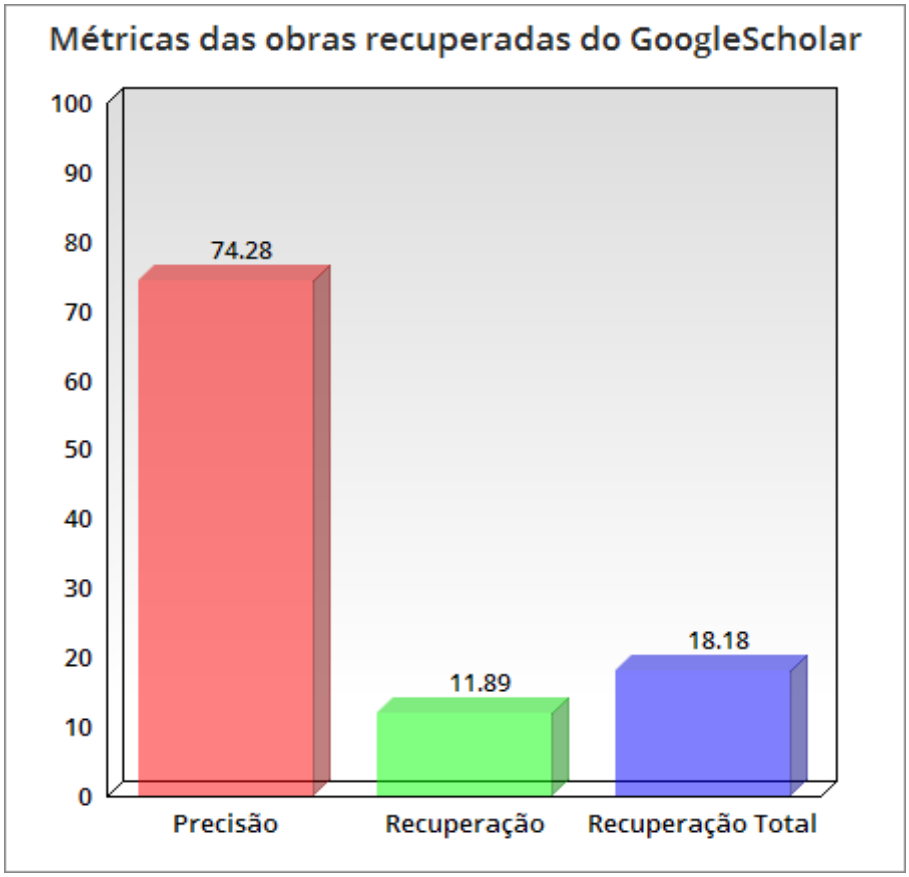


Figura 6. Resultado dos testes realizados com as obras do GoogleScholar

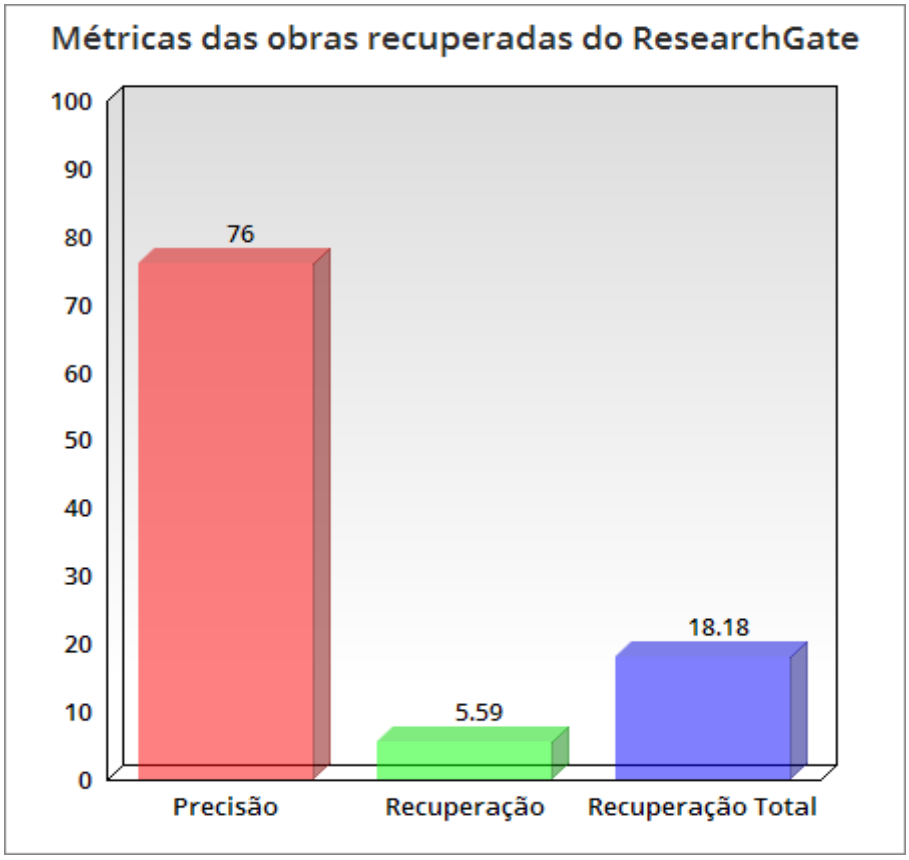


Figura 7. Resultado dos testes realizados com as obras do ResearchGate

5. CONSIDERAÇÕES FINAIS

Através da realização dos experimentos é possível concluir que a ferramenta atingiu o seu objetivo. 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 dados recuperados de forma unificada.

Apesar de a ferramenta obter êxito nos objetivos almejados ela ainda apresenta vários pontos que podem ser melhorados de forma significativa e permite o suporte a criação de ideias que não foram desenvolvidas. A seguir são listadas sugestões de melhorias e ideias que visam melhorar a qualidade da aplicação.

1. Melhorias no algoritmo de comparação de obras.
2. Incrementar o número de repositórios avaliados para extração e comparação de dados.
3. Informar os pesquisadores que tiveram o currículo avaliado sobre as obras encontradas.
4. Otimização do tempo de acesso aos repositórios no uso de servidores de proxy.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] CNPQ. Sobre a Plataforma. Disponível em <http://lattes.cnpq.br/web/plataforma-lattes/home>. Acesso em: 24 de novembro de 2018.

[2] REPANOVICI, Angela. Measuring the visibility of the university's scientific production through scientometric methods: An exploratory study at the Transilvania University of Brasov, Romania. Performance Measurement and Metrics. v.12, n.2, pp.106-117, 2011.

[3] PINKERTON, Brian. WebCrawler: Finding What People Want, Dissertação (Doutorado) - University of Washington, USA, 2000.

[4] GEUNA, Aldo et al. SiSOB data extraction and codification: A tool to analyze scientific careers. Research Policy. v.44, n.9, pp.1645-1658, 2015.

[5] WU, Jian et al. CiteSeerX: AI in a Digital Library Search Engine. AI Magazine. v.36, pp.35-48, 2014.