OntoScraping: Enriquecimento de Ontologias por meio de *Web Scraping*

OntoScraping: Enriching Ontologies Using Web Scraping

Amanda Tamires Inacio Silva[1]

Resumo

O presente artigo apresenta o OntoScraping, uma solução baseada em *Web Scraping* que objetiva enriquecer os metadados da OntoExpLine, uma ontologia voltada para *workflows* científicos. O programa foi implementado na linguagem Python, utilizando o *framework Scrapy.* O trabalho aborda conceitos importantes para compreensão do projeto, bem como detalhes relacionados ao desenvolvimento, composição dos módulos criados, ferramentas utilizadas e exemplos de seu funcionamento.

Palavras-chave: Web Scraping. Ontologias. Workflows Científicos.Scrapy.

Abstract

This paper presents OntoScraping, a Web Scraping-based solution that aims to enrich the metadata of OntoExpLine, an ontology for scientific workflows. The program was implemented in Python language, using the Scrapy framework. The work adderesses important concepts are presented for understanding the project, as well as details related to the development, composition of the modules created, tools used and examples of its operation.

Keywords: Web Scraping. Ontology. Scientific Workflow.Scrapy.

Aprovado em: 14/05/2021 Versão Final em: 17/05/2021

[1]Trabalho de conclusão de curso apresentado ao curso de Bacharelado em Sistemas de Informação da Universidade Federal Fluminense como requisito parcial para conclusão do curso.

Amanda Tamires Inacio Silva- UFF; <amandainacio@id.uff.br>

INTRODUÇÃO

Com a crescente expansão da Internet, atualmente é possível encontrar quase todo tipo de informação sobre os mais diversos temas. A variedade dos dados disponíveis evidencia o sucesso da rede mundial de computadores. Porém, o volume excessivo de dados não estruturados que se tem formado no ambiente da web por vezes, ao invés de ajudar, acaba dificultando a busca.

> A necessidade de se obter informação de interesse relativamente a determinado tema, mas tendo que recorrer a diversas fontes para obter toda a informação que pretende obter e comparar, torna-se um processo moroso para o utilizador. (BARREIRA**,** 2014, p.3)

A partir desta constatação, podemos observar o surgimento de uma demanda por ferramentas que facilitem o processo de obtenção de informações completas e estruturadas. Neste contexto, a automatização da extração e do tratamento dos dados surge como sendo uma boa alternativa, podendo ser utilizada, por exemplo, a técnica de coleta de dados na *web, também* conhecida como *Web Scraping.*

Além da captura dos dados com o intuito de convertê-los em informação útil, também é necessário que haja uma preocupação acerca da representação de tais dados de forma que possibilite a construção de conhecimento. Ou seja, como correlacionar as informações obtidas para que façam sentido dentro de um domínio específico. Uma solução possível para esta questão é o uso de ontologias, pois explicitam a relação entre conceitos (Guarino, 2009).

O presente artigo tem como objetivo apresentar o OntoScraping, uma solução voltada para enriquecimento de uma ontologia, chamada OntoExpLine, através do processo *Web Scraping*. Serão apresentados detalhes a respeito da implementação, definição de conceitos necessários à compreensão do trabalho, bem como um exemplo de funcionamento da solução criada.

2- REFERENCIAL TEÓRICO

2.1- *Web Scraping*

A técnica de *Web Scraping* pode ser definida como uma coleta de dados automatizada a partir de sites disponíveis na web. São utilizados scripts e programas para extrair os dados, que poderão ser armazenados em arquivos ou bancos de dados para serem analisados e utilizados das mais diversas formas. O processo também é conhecido como raspagem de

dados da web. Para Mitchell (2016, p. 2), "*Web Scraping* é a prática de coletar dados por qualquer meio que não seja um programa interagindo com uma API (ou, obviamente, por um ser humano usando um navegador web)".

Apesar do termo *Web Scraping*, consensualmente, referir-se à extração de dados feita de maneira automática, também é possível realizá-la manualmente; porém, quando o volume de informações a serem analisadas é muito grande, o processo pode tornar-se lento e custoso para o usuário. Por isso, a automatização deste trabalho tem sido amplamente utilizada, subsidiando inclusive a tomada de decisões em organizações. Segundo Pacheco, colunista do blog *Geek Hunter*, advogados, empresas de marketing, empresas de mídia e jornalistas estão incluídos no público interessado pela utilização da técnica. Redes sociais como Twitter e Facebook, por exemplo, realizam *Web Scraping* para verificar os assuntos e tópicos que estão em alta.

Um script de raspagem de dados pode ser implementado em diversas linguagens de programação. O OntoScraping foi desenvolvido em Python que tem se destacado por ser uma linguagem de fácil aprendizado e por possuir diversos frameworks e bibliotecas voltados para *Web Scraping*.

O processo de coleta deve ser iniciado com uma análise do site que será fonte da extração, sendo necessário entender como ele está estruturado e como suas URLs[2] são construídas. Em seguida, deverá ser criado um script cujo funcionamento, resumidamente, consiste em fazer uma requisição HTTP[3], a partir da URL informada, receber a resposta do servidor e armazenar os dados em um objeto Python. De posse do conteúdo HTML[4] resultante da requisição, utilizando os recursos disponíveis, é possível separar as informações relevantes dos elementos HTML e extrair o que de fato importa para finalidade do seu projeto.

Nas Seções 3 e 4 serão detalhadas as ferramentas abordadas durante a criação do OntoScraping, que será utilizado para enriquecimento da ontologia OntoExpLine.

2.2- Ontologias

Para Gruber (1993), "uma ontologia pode ser entendida como uma especificação explícita de uma conceitualização". Através do uso de ontologias é possível representar a relação entre indivíduos, conceitos e atributos de acordo com as regras necessárias para que o relacionamento entre eles faça sentido dentro de um domínio específico.

[2] URL: Endereço virtual de uma página ou web site.
[3] HTTP: Protocolo de Transferência de Hipertexto.
[4] HTML: Linguagem de Marcação de Hipertexto.

As ontologias podem ser representadas formalmente, ou seja, direcionadas à interpretação computacional, ou graficamente dedicadas ao entendimento humano. Para representação formal, existem algumas linguagens que podem ser utilizadas, como por exemplo, a OWL[5] (ISOTANI, 2015).

Existem vários tipos de ontologias, dentre essas, podemos destacar a ontologia superior, a de domínio e a de tarefas. Através da ontologia superior é possível descrever termos de vários domínios diferentes. Ela contém definições gerais que podem ser aplicadas em diversos contextos. Já a de domínio, possui dados relativos a um domínio específico, possibilitando a representação de conceitos dentro de um escopo definido. Através da ontologia de tarefa é possível utilizar o conhecimento contido em ontologias de domínio para solucionar problemas diversos. A OntoExpLine que será abordada na próxima subseção é considerada uma ontologia de tarefa.

2.3- OntoExpLine

Experimentos de Ciência Computacional e Engenharia (CSE), em alguns casos, são implementados como *workflows* científicos[6]. Que são compostos por uma sequência de atividades. Os *workflows* considerados para o contexto deste trabalho são voltados para análise *in silico*, ou seja, baseados em simulação computacional. Com isso, a maior parte das atividades que os compõem são implementadas por *softwares*. Cada experimento pode envolver uma combinação variada de dados, parâmetros de entrada e atividades, originando variações de *workflows*. Porém, os Sistemas de Gerenciamento de W*orkflows* Científicos (SGWfC) não estão preparados para permitir a associação de *workflows* diferentes a um mesmo experimento. A impossibilidade de se registrarem as diversas maneiras possíveis de implementação de um mesmo ensaio podem gerar retrabalho para os cientistas e favorecer a ocorrência de falhas. De acordo com Dias et. al. (2020, p.2), uma solução para este problema seria o uso Linhas de Experimento Algébricas (LinExp).

> O conceito de LinExp objetiva auxiliar o cientista no processo de derivação de *workflows* executáveis a partir de uma definição abstrata. Uma LinExp modela todas as possíveis variações de programas que implementam uma atividade, quais atividades são opcionais, seus parâmetros, dependências e restrições encontradas em um determinado experimento de CSE. (Dias et. al, 2020, p. 2)

[5] Definição de OWL: https://www.w3.org/OWL/- <acessado em: 27/04/2021>

[6] Segundo Dias et. al. (2020), *workflows* científicos podem ser definidos como uma abstração que representa as etapas de um experimento de CSE.

Apesar de uma LinExp modelar os possíveis *workflows* diferentes para um experimento, em sua versão atual, não existe suporte para armazenar informações detalhadas a respeito de cada atividade que compõe uma derivação. Essa limitação de metadados faz com que a LinExp não seja suficiente para validar a modelagem de todas as possibilidades de estruturação de um experimento científico, sendo necessária a sua conexão com ontologias, através das quais, de acordo com Dias et. al. (2020), será "possível instanciar, caracterizar e relacionar programas e atividades abstratas, agregando metadados relativos à sua execução, dependências de software, dados do domínio de aplicação (e.g., bioinformática) e recursos"

Neste contexto, surge a OntoExpLine, que é uma ontologia de tarefa voltada para LinExp. Ela foi criada através da abordagem SABiO[7] e seu objetivo é apoiar os cientistas durante a modelagem de seus *workflows,* provendo a geração automática de "versões variantes validadas de um mesmo *workflow"* (Dias et al., 2020, p.7).

A arquitetura da OntoExpLine é composta por quatro módulos: um módulo superior baseado na abordagem de LinExp voltado à caracterização abstrata de linhas de experimento (como atividades e itens de dado consumidos e gerados); um módulo de tarefa, onde são definidos componentes concretos que compõem o *workflow* (como programas e entidades de dados); um módulo de domínio utilizado para correlacionar conhecimento de operações de domínio aos elementos abstratos e concretos do *workflow*; além de um módulo de metadados. Os módulos de domínio e metadados são totalmente flexíveis, podendo ser substituídos de acordo com a necessidade do *workflow*, já os módulos de LinExp e de tarefa são pré-estabelecidos pelo modelo proposto por Ogasawara e pela ontologia ProvONE[8], respectivamente.

Apesar de possuir uma estrutura robusta, OntoExpLine não possui mecanismo automático de definição de indivíduos, provendo apenas o modelo de dados e arquivo OWL. Dessa forma, a estrutura do *workflow*, tanto em sua forma abstrata quanto na sua forma concreta, deve ser realizada manualmente pelo cientista, o que é justificável principalmente pelas particularidades de cada experimento.

Ademais, a definição de elementos de metadados também deve ser feita de forma manual, e tal atividade pode inviabilizar o uso do módulo, pois, dependendo da quantidade de elementos concretos especificados tal etapa acaba se tornando custosa. O não uso do módulo

[7] Systematic Approach for Building Ontologies

[8] http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html

de metadados pode, por exemplo, inviabilizar o reuso do *workflow* especificado na ontologia, visto que informações sobre dados e configurações não estarão disponíveis.

Com base nisso percebeu-se que, uma vez definida a estrutura concreta do experimento, a tarefa de captura e definição de metadados pode ser automatizada, visto que indivíduos que representam programas e seus relacionamentos já foram instanciados pelo cientista.

O OntoScraping foi criado para enriquecer a OntoExpLine buscando na web, de maneira automatizada, informações complementares sobre programas utilizados por *workflows* científicos que compõem as linhas de experimento. Um dado importante que pode ser útil para o cientista, é a versão do *software*, por exemplo. Afinal, a compatibilidade dos programas com os experimentos poderá ser comprometida dependendo das mudanças implementadas durante a atualização de uma versão.

3- OntoScraping

O processo de criação do OntoScraping se iniciou com a escolha da fonte de dados e com a definição das informações que seriam relevantes para o enriquecimento da ontologia. Nesta etapa os criadores da OntoExpLine definiram que, para este momento, seria suficiente capturar dados constantes nas páginas dos programas na *Wikipedia.* A partir de então, foi feita uma análise dessas páginas a fim de verificar quais informações seriam extraídas e incorporadas aos metadados.

Durante a verificação das páginas, foi identificada uma tabela que continha um grupo de atributos, relacionados aos programas, que seriam adequados para extração. Tais informações variam de acordo com o software mas, na maioria dos casos, se mostraram suficientemente relevantes para o propósito do trabalho.

O programa foi desenvolvido em Python, linguagem escolhida por fornecer diversas bibliotecas e *frameworks* que dão suporte eficiente à implementação de *Web Scraping*, tais como *Beatiful Soup*[9]*, Scrapy*[10]*, Mechanize*[11], entre outros.

O projeto possui dois módulos principais, um deles se comunica com a OntoExpLine para gerar a URL do programa requerido e armazenar as informações, o outro realiza a coleta

[9] Beautiful Soup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ <acessado em 05/05/2021>
[10] Scrapy: https://docs.scrapy.org/en/latest/ <acessado em 05/05/2021>
[11] Mechanize: https://mechanize.readthedocs.io/en/latest/ <acessado em 05/05/2021>

dos dados. A implementação do módulo de coleta foi feita através do *framework Scrapy*, que pode ser utilizado para o propósito de extração de dados, mas que também possui suporte para processamento de informações e mineração de dados[12] .

O *Scrapy* tem como elemento principal de sua arquitetura os *Spiders* , que são classes onde devemos definir as diretrizes que nortearão o rastreamento e a análise das páginas que serão objeto da captura de dados. Os *Spiders* funcionam, resumidamente, da seguinte forma: como entrada eles recebem uma ou mais URLs; em seguida é feita uma requisição para estas URLs; a resposta dessa requisição (o conteúdo HTML da página) é recebida pelo método *parse* (criado pelo desenvolvedor, no corpo do *Spider*) e então, de acordo com os parâmetros definidos, os dados são tratados e retornados ao usuário. A partir daí é possível salvar as informações em arquivos do tipo JSON[13] ou XML[14] , por exemplo, ou ainda persisti-las em um banco de dados[15].

Além da simplicidade de uso, o *Scrapy* fornece um desempenho muito satisfatório, pois ele consegue gerenciar diversas requisições através do agendamento e do processamento assíncrono, ou seja, ele pode continuar enviando solicitações ou realizar outra tarefa enquanto os resultados estão sendo gerados[16] . Com isso, as respostas são obtidas com maior rapidez. Abaixo, uma visão geral da arquitetura do *framework Scrapy* e um esboço do seu fluxo de dados.

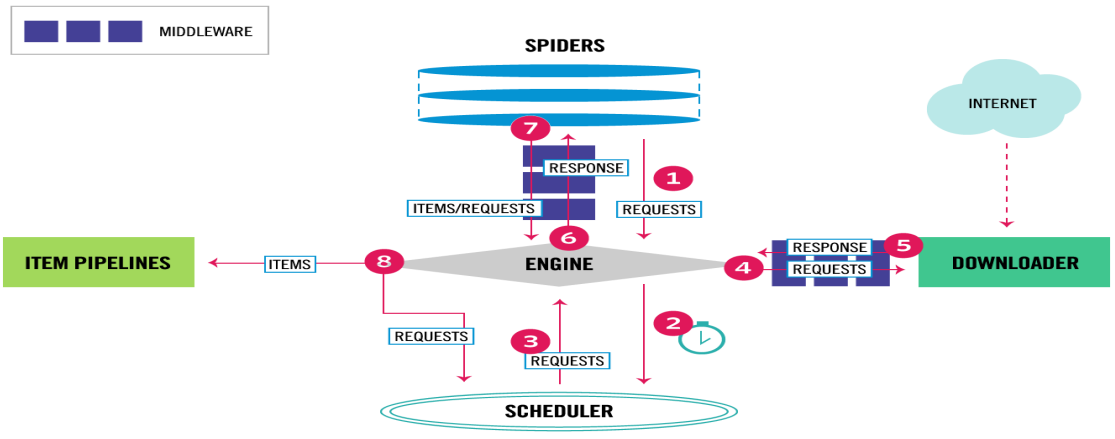


Figura 1. Diagrama de Arquitetura do Scrapy.
Fonte: https://docs.scrapy.org/en/latest/topics/architecture.html

[12] https://docs.scrapy.org/en/latest/intro/overview.html<acessado em 05/05/2021>
[13] https://www.json.org/json-en.html <acessado em 05/05/2021>
[14] https://www.w3.org/TR/REC-xml/ <acessado em 05/05/2021>
[15] Spiders: https://docs.scrapy.org/en/latest/topics/spiders.html <acessado em 05/05/2021>
[16] https://docs.scrapy.org/en/latest/intro/overview.html<acessado em 05/05/2021>

Fluxo de Dados[17]:

1) O *Engine* recebe as requisições iniciais do *Spider*;
2) O *Engine* agenda as requisições no *Scheduler* e solicita as próximas requisições;
3) O *Scheduler* retorna as próximas requisições para o *Engine;*
4) O *Engine* envia requisições para o *Downloader* através dos *Middleware* do *Downloader (process_request());*
5) Quando a página termina o *download*, o *Downloader* gera a resposta (daquela página) e a envia para o *Engine,* pelo *Middleware* do *Downloader (process_response());*
6) O *Engine* recebe a resposta do *Downloader* e envia, via *Middleware Spider (process_spider_input),* para o *Spider* processar;
7) O *Spider* processa a resposta e retorna os itens processados e as novas requisições para o *Engine,* através do *Middleware Spider (process_spider_output);*
8) O *Engine* envia os itens processados para o *Item Pipeline* e, em seguida, envia requisições processadas para o *Scheduler* e solicita as próximas requisições;
9) O processo se repete a partir da etapa 1, até que não haja mais requisições no *Scheduler.*

O OntoScraping possui um *Spider (*WikiScraping*)* que recebe uma ou mais URLs, e para cada página retornada extrai o nome do programa e sua descrição, em seguida, busca a tabela de dados do *software*, percorrendo cada linha e verificando, com suporte da biblioteca *re*[18] do *Python* (operações com expressão regulares), os atributos disponíveis e guardando cada um deles no dicionário[19] correspondente. Ao final, as informações coletadas são retornadas para utilização na OntoExpLine.

Após a criação do módulo de coleta de dados, foi realizada uma análise da solução pelos criadores da OntoExpLine, que resultou na identificação da possibilidade de implementação de melhorias, tais como, alteração na forma como a execução do *Spider* era disparada (ocorria através do envio de um comando no terminal) e inclusão da possibilidade de busca da página do programa pelo nome e não apenas pela URL.

Para incorporar as modificações sugeridas ao projeto, foi criado um novo módulo (ontoscrapy.py) dedicado à comunicação entre a OntoExpLine e o módulo de coleta. Ele

[17] https://docs.scrapy.org/en/latest/topics/architecture.html <acessado em 06/05/2021>
[18] https://docs.python.org/pt-br/3/library/re.html <acessado em 05/05/2021>
[19] https://docs.python.org/pt-br/3/tutorial/datastructures.html <acessado em 05/05/2021>

possui duas funções (*generate_url e search_metadata*), uma delas recebe como parâmetro o nome do programa e retorna a sua URL na *Wikipedia.* A outra função tem o objetivo alterar o campo *start_urls*[20] do *Spider* com a URL do programa desejado. Ainda neste módulo, com o suporte da biblioteca *OS*[21] do *Python* ocorre a chamada de execução do *Spider* (já modificado com a URL desejada). Abaixo, o diagrama que apresenta, de forma resumida, o funcionamento do OntoScraping e sua relação com a OntoExpLine.

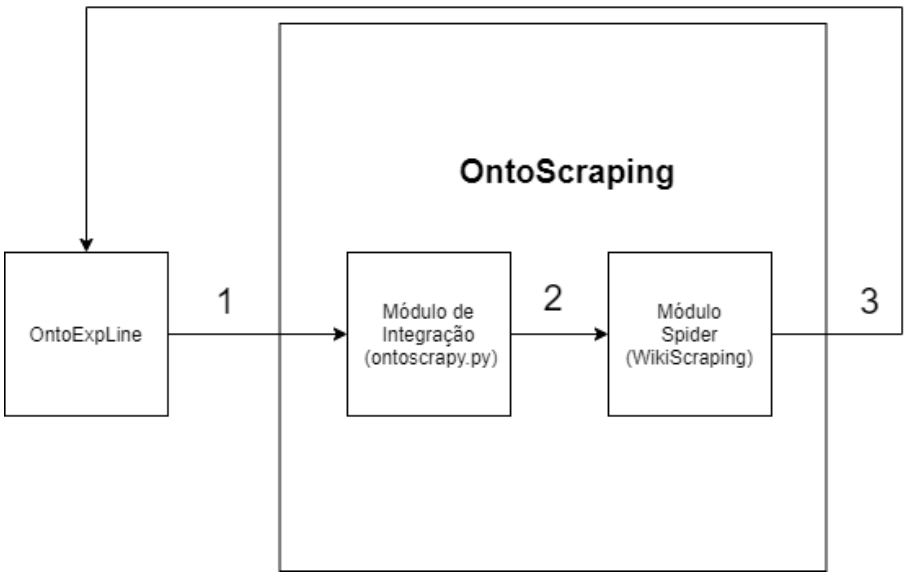


Figura 2. Diagrama de funcionamento do OntoScraping.

Fonte: Criado pela própria autora.

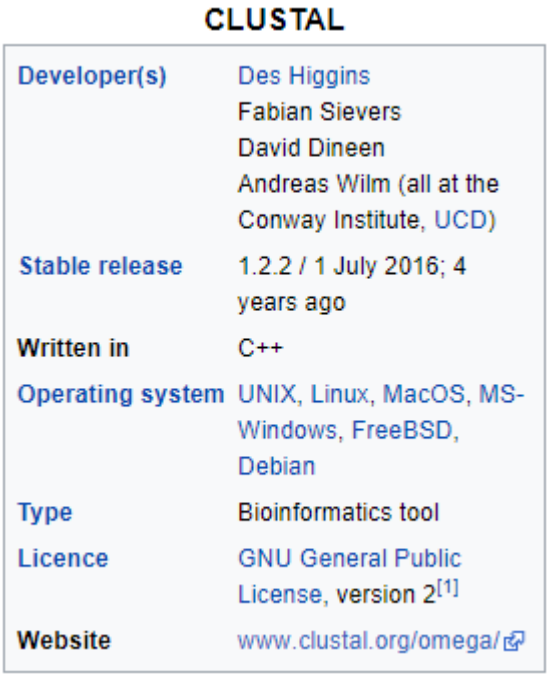
Fluxo:

1) A OntoExpLine envia para o módulo de integração o nome do programa a ser buscado;
2) O módulo de integração, através da função *generate_url,* gera a URL do programa; em seguida, a função *search_metadata* altera o *Spider* com a URL gerada; por fim, é feita a chamada do Spider modificado utilizando a biblioteca OS;
3) O *Spider* realiza a raspagem e limpeza dos dados e retorna as informações para OntoExpLine, para que sejam salvas diretamente na ontologia.

[20] https://docs.scrapy.org/en/latest/topics/spiders.html?highlight=START_URLS#scrapy.spiders.Spider.start_urls <acessado em 06/05/2021>
[21] https://docs.python.org/pt-br/3/tutorial/stdlib.html <acessado em 06/05/2021>

4- Avaliação do OntoScraping

O processo de criação do OntoScraping foi realizado após uma análise de diversas páginas de *softwares* que poderiam ser objeto da extração. Durante este processo foi identificada uma variação entre as estruturas das páginas, o que poderia inviabilizar a recuperação dos dados. Porém, observou-se que, na maioria delas, havia uma tabela com informações sobre o programa, a partir de então, a coleta se baseou na extração dos dados desta tabela. Abaixo, um exemplo da referida estrutura.



CLUSTAL

Developer(s)	Des Higgins Fabian Sievers David Dineen Andreas Wilm (all at the Conway Institute, UCD)
Stable release	1.2.2 / 1 July 2016; 4 years ago
Written in	C++
Operating system	UNIX, Linux, MacOS, MS-Windows, FreeBSD, Debian
Type	Bioinformatics tool
Licence	GNU General Public License, version 2[1]
Website	www.clustal.org/omega/

Figura 3. Tabela retirada da página do programa Clustal.
Fonte:https://en.wikipedia.org/wiki/Clustal

Como os atributos disponíveis variam de *software* para *software,* foi realizado um mapeamento para identificar tal variação. Com isso, a lógica da extração dos dados passou a ser baseada na comparação das informações constantes em cada linha da tabela com uma lista que, em cada posição, contém um atributo mapeado. Foram identificados os seguintes possíveis atributos: Developer, Repository, Write In, Operating System, Type, License, Website, Stable Release, Original Author(s), Initial Release, Plataform, Available In. Para tornar a captura mais completa e cobrir os casos em que os atributos não sejam encontrados, além destes dados, inicialmente são salvos o nome do programa e a sua descrição, por se tratarem de itens que estão sempre disponíveis nas páginas da *Wikipedia*.

A avaliação do OntoScraping foi baseada no estudo de caso realizado por Dias et. al. (2020, p.6), que aborda o *workflow SciPhy,* que "tem como objetivo construir árvores filogenéticas a partir de sequências de DNA, RNA e aminoácidos" (Dias, et. al. 2020, p.6). Este *workflow* se mostra suficiente para a análise de exemplo, pois de acordo com Dias et. al. (2020), em seu fluxo de dados existem atividades que podem ser implementadas por programas diferentes. O que significa que o cientista precisará realizar buscas sobre os *softwares* para escolher o que melhor se aplica ao seu objetivo específico, e tal busca pode ser muito custosa para o experimento. Dias et. al.(2020) destaca que esta escolha é de suma importância, pois pode impactar no desempenho do *workflow*, visto que cada programa pode consumir uma quantidade de recursos computacionais diferentes. Por isso, a OntoExpLine se propõe a reunir metadados com o objetivo de que, a partir da definição concreta do CSE, eles estejam disponíveis de maneira acessível para auxiliar os cientistas na modelagem de seus *workflows*.

Partindo deste princípio, nesta avaliação será demonstrado o uso do OntoScraping para capturar e salvar na OntoExpLine metadados de um dos possíveis programas que podem ser utilizados em atividades do *workflow SciPhy.* Cabe ressaltar que este processo de coleta terá efeito de enriquecimento da ontologia, visto que, ela já possui diversos metadados registrados.

De acordo com Dias et. al. (2020, p.6), o *workflow SciPhy* é composto por cinco atividades abstratas, cada uma delas pode ser implementada por um ou mais programas. Para esta análise, abordaremos a atividade abstrata número 2 que pode ser executada por três programas diferentes: *Muscle*[22]*, Mafft*[23] ou *ClustalW*[24]. Para o nosso exemplo utilizaremos o *Mafft*. A seguir, será apresentado o diagrama de funcionamento do OntoScraping ilustrando o processo de coleta de dados do *Mafft*, bem como, detalhes do fluxo de dados desta etapa.

[22] https://en.wikipedia.org/wiki/MUSCLE_(alignment_software)
[23] https://en.wikipedia.org/wiki/MAFFT
[24] https://en.wikipedia.org/wiki/Clustal

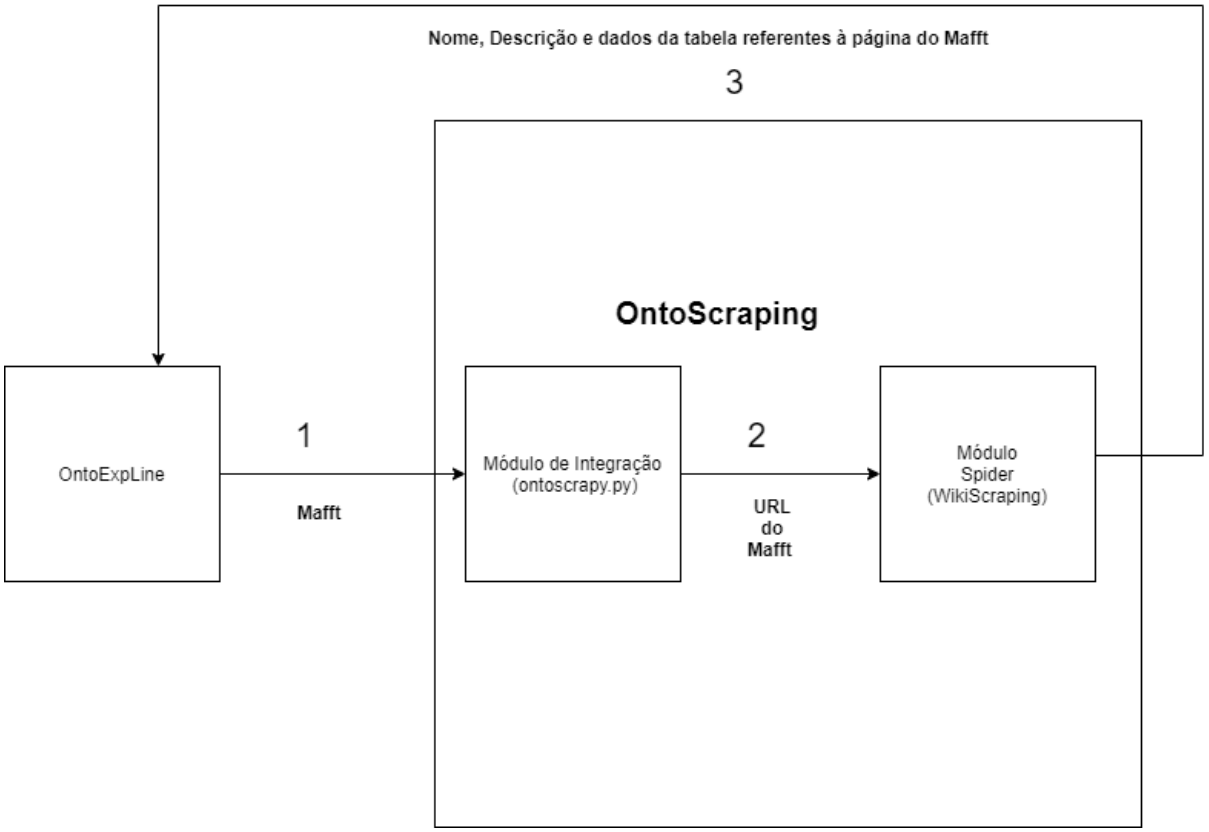


Figura 4. Diagrama de funcionamento do OntoScraping (Mafft).

Fonte: Criado pela própria autora.

Fluxo de dados:

1. A OntoExpLine envia o nome do programa “Mafft” para o módulo de integração;
2. O módulo de integração através da função *generate_url* gera a seguintes URL: https://en.wikipedia.org/wiki/MAFFT ; em seguida, a função *search_metadata* altera o *Spider* com a URL gerada; por fim, é feita a chamada do *Spider* modificado utilizando a biblioteca OS;
3. O *Spider* realiza a raspagem e limpeza dos dados, retornando as seguintes informações para OntoExpLine:

   Mafft:

   {"name_program": "MAFFT - Wikipedia"},

{"description": "In bioinformatics, MAFFT (for multiple alignment using fast Fourier transform) is a program used to create multiple sequence alignments of amino acid or nucleotide sequences. Published in 2002, the first version of MAFFT used an algorithm based on progressive alignment, in which the sequences were clustered with the help of the Fast Fourier Transform.[2] Subsequent versions of MAFFT have added other algorithms and modes of operation,[3] including options for faster alignment of large numbers of sequences,[4] higher accuracy alignments,[5] alignment of non-coding RNA sequences,[6] and the addition of new sequences to existing alignments.[7]\n"},

{"developer": "Developer(s) Kazutaka Katoh"},

{"release": "Stable release 7.475\n / 23\u00a0November 2020; 5 months ago\u00a0(2020-11-23)"},

{"written_in": "Written in C"},

{"operating_system": "Operating system UNIX, Linux, Mac, MS-Windows"},

{"category": "Type Bioinformatics tool"},

{"license": "Licence BSD [1]"},

{"website": "Websitemafft.cbrc.jp/alignment/software/"}

A partir de então, as informações coletadas são salvas diretamente na ontologia no módulo dedicado aos metadados. A seguir serão demonstradas formas de visualização dos metadados já salvos na OntoExpLine, após o processo de extração realizado pelo OntoScraping.

Na figura abaixo é possível visualizar a relação entre o super conceito “Metadata” e os diversos sub conceitos, que são os tipos de metadados que a OntoExpLine pode relacionar com a classe programa.

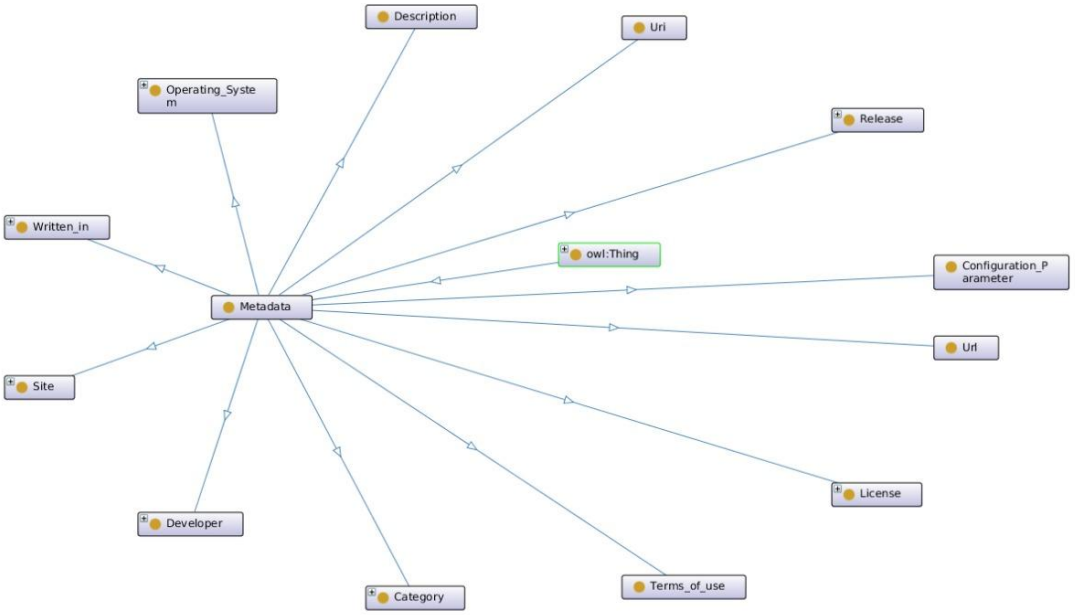


Figura 5. Ramo de metadados da OntoExpLine.

Fonte: OntoExpLine

Na próxima figura, ainda no ramo de metadados, conseguimos visualizar a relação dos indivíduos extraídos pelo OntoScraping com os conceitos existentes.

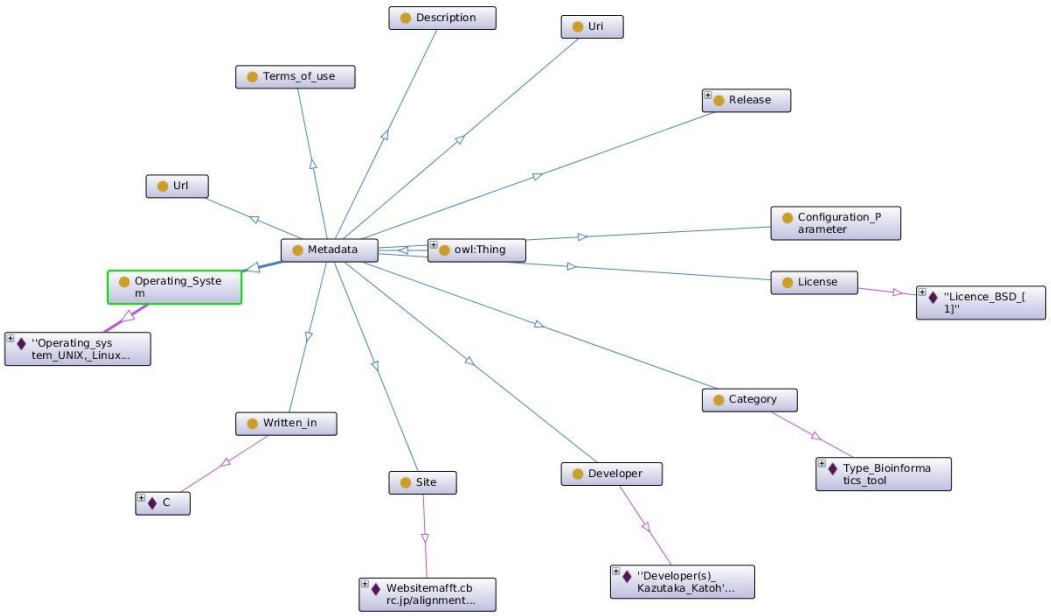


Figura 6. Ramo de metadados, relação entre conceitos e indivíduos.

Fonte: OntoExpLine

Na imagem abaixo podemos observar o programa *Mafft*, sua relação com o super conceito "*Program*" e com os metadados extraídos pelo OntoScraping.

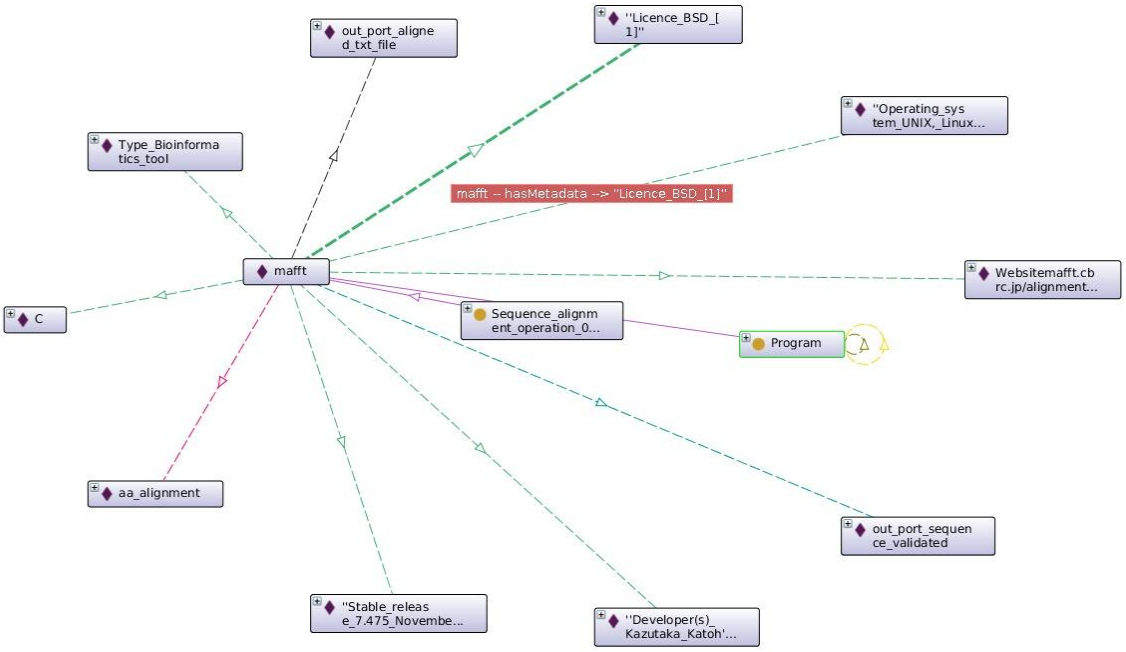


Figura 7. Ramo *Program,* relação com Mafft e seus metadados.

Fonte: OntoExpLine

Os dados extraídos poderão ser úteis para auxiliar os cientistas no processo decisório. Especialmente a versão do software, bem como o sistema operacional, pois são informações muito relevantes para avaliar a compatibilidade do programa com o experimento. Com relação a versão, por exemplo, deve-se destacar que uma atualização pode trazer mudanças que impactem a execução da atividade ou até mesmo inviabilizem sua conclusão. O tipo de licença também é considerado de grande relevância, afinal é necessário estar atento aos limites de uso de cada categoria para que as regras estabelecidas sejam respeitadas.

5-CONCLUSÃO

Realizar buscas na web pode ser considerado algo trivial, afinal, hoje em dia a maioria das pessoas possui acesso à internet. Porém, quando a pesquisa precisa ser feita em diversas páginas diferentes ou o volume de informações a serem analisadas é muito grande, tal processo pode se tornar lento e custoso. Partindo deste princípio, o uso do *Web Scraping* para automatizar a captura de dados da web tem sido amplamente utilizado pelas mais diversas

áreas da sociedade. Porém, para alguns propósitos, a captura por si só nem sempre é suficiente. É necessário que as informações estejam relacionadas de forma a fazerem sentido dentro de um contexto. E para esta finalidade, uma alternativa é o uso de ontologias, através das quais é possível correlacionar conceitos, atributos e indivíduos.

As sessões anteriores deste artigo apresentaram o OntoScraping, um programa de raspagem de dados da web que foi criado para enriquecer o módulo de metadados da OntoExpLine, uma ontologia voltada para LinExp. Através da OntoExpLine é possível modelar as diferentes formas de execução de *workflows* científicos. O OntoScraping atua capturando dados de programas utilizados nestes *workflows,* para que sejam armazenados na ontologia e disponibilizados para os cientistas, subsidiando a tomada de decisões durante a modelagem dos *workflows.*

Durante a execução deste trabalho foram utilizados diversos conhecimentos adquiridos ao longo da graduação, principalmente assuntos relacionados às disciplinas de Programação, Desenvolvimento Web e Representação da Informação. Com relação à possível continuidade do projeto realizado, pode-se considerar a inclusão da busca a outros dados relevantes sobre os *softwares,* bem como, a pesquisa através de outras fontes, além da *Wikipedia*.

REFERÊNCIAS

DIAS, Luiz Gustavo; LOPES, Bruno; DE OLIVEIRA, Daniel. OntoExpLine: Rumo a uma Ontologia para Representação de Linhas de Experimento Algébricas. In: **Anais do XIV Brazilian e-Science Workshop**. SBC, 2020. p. 33-40.

GRUBER, Thomas R. A translation approach to portable ontology specifications. **Knowledge acquisition**, v. 5, n. 2, p. 199-220, 1993.

GUARINO, Nicola; OBERLE, Daniel; STAAB, Steffen. What is an ontology?. In: **Handbook on ontologies**. Springer, Berlin, Heidelberg, 2009. p. 1-17.

ISOTANI, Seiji; BITTENCOURT, Ig Ibert. **Dados Abertos Conectados: Em busca da Web do Conhecimento**. Novatec Editora, 2015.

MITCHELL, R. Web Scraping com Python. São Paulo: Novatec, 2016.

OGASAWARA, Eduardo et al. Experiment line: Software reuse in scientific workflows. In: **International Conference on Scientific and Statistical Database Management**. Springer, Berlin, Heidelberg, 2009. p. 264-272.

PACHECO, Renato Anhaia Flud, Como fazer web scraping python de maneira fácil e rápida. **Blog Geek Hunter**, 13 de Março de 2020. Disponível em: <https://blog.geekhunter.com.br/como-fazer-um-web-scraping-python/#:~:text=Web%20scraping%20%C3%A9%20uma%20coleta,mercado%2C%20entre%20muitas%20outras%20coisas > Acesso em: 25 de abril de 2021.

.

APÊNDICE A – Lista de ilustrações