Submission date: 11/09/2023
Camera ready submission date: 04/06/2024

1st round notification: 14/05/2024
Available online: 13/07/2024
Publication date: 13/07/2024

Section: Regular article

Coleta de Dados Turísticos com Web Scraping: Problemas, Soluções e Otimizações

Tourist Data Collection With Web Scraping: Problems, Solutions and Optimizations

João Pedro Martins de Paula [1], **João Eugenio Marynowski** [1], **José Elmar Feger** [1]

[1]Setor de Educação Profissional e Tecnológica (SEPT) – Universidade Federal do Paraná (UFPR)
Curitiba, Paraná – Brasil

{joaopaula,jeugenio,elmar}@ufpr.br

Abstract. *The Internet can be considered one of the biggest data sources for humanity, but it's use is intended for human readers, not for automatic reading. Internet automated data extraction faces some obstacles involving data arrangement and the frequent updates that websites suffer. This study analyzes the extraction of tourist data for later categorization, listing faced problems, like page navigation and program performance, and presenting their solutions. Scraping algorithms are presented based on a general process and the different page-loading method, exploring them in a case study. Are evaluated the use of computational resources, the scraping rate and the security impacts, modifying the implementation accordingly to the problems that arise in the processing and documenting the solutions and optimizations, e.g. avoiding the Denial of Service (DoS) and applying parallel programming (multithreading). The data and scripts generated are available in a public repository for reproduction and inspiration.*

Keywords. *Web Scraping; Data Extraction; Selenium; Tripadvisor; Data acquisition; Comments.*

Resumo. *A Internet pode ser considerada uma das maiores fontes de dados que a humanidade possui acesso, mas esse acesso está otimizado para a leitura humana e não para a leitura automatizada. A extração de dados da Internet de forma automatizada enfrenta diversos obstáculos envolvendo a disposição do conteúdo e a atualização frequente da estrutura onde os dados estão dispostos. Este estudo analisa a extração de dados turísticos para posterior categorização, listando os problemas enfrentados, como o avanço de páginas e a eficiência do programa, e as soluções consideradas para resolvê-los. São apresentados algoritmos baseados no processo geral de extração e nos diferentes métodos de*

Cite as: de Paula, J.P.M., Marynowski, J. E., & Feger, J. E. (2024). Tourist Data Collection With Web Scraping: Problems, Solutions and Optimizations (Coleta de Dados Turísticos com Web Scraping: Problemas, Soluções e Otimizações). iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), Vol 17(1), 8:1-8:19. DOI: 10.5753/isys.2024.3644

carregamento de páginas, explorando-os no estudo de caso. São avaliados o uso de recursos computacionais, a taxa de extração e impactos na segurança, modificando a implementação de acordo com os problemas ocorridos no processamento e documentando as soluções e otimizações aplicadas, como evitar a negação de serviço (Denial of Service - DoS) e o emprego de programação concorrente (multithreading). Os dados e scripts gerados estão disponíveis em repositório público para reprodução e inspiração.

Palavras-Chave. *Web Scraping; Extração de dados; Selenium; Tripadvisor; Aquisição de dados; Comentários.*

1. Introdução

A grande quantidade de dados existentes na Internet tem o potencial de alavancar pesquisas científicas baseadas nas experiências e opiniões dos usuários [Vieira and Moura 2020], bastando ao pesquisador utilizar técnicas automatizadas, de preferência, para a extração dos dados online. A técnica automatizada de extração de dados mais utilizada pela comunidade científica é o Web Scraping [Zhao 2017, Diouf et al. 2019].

Infelizmente, os dados na Internet nem sempre estão dispostos de maneira a facilitar a sua coleta. Isso se deve a vários motivos, como o de que a intenção inicial ao expor os dados na web não era de que fossem usados cientificamente, mas apenas para consulta casual [Oliveira et al. 2020]. Isso afeta a forma como estão dispostos, favorecendo a leitura visual em detrimento da extração automatizada. Outros websites, ainda, podem até mesmo querer dificultar a extração de seus dados de forma automática, usualmente feita pela concorrência, e assim acabam afetando também a pesquisa científica.

Por ser uma técnica bem disseminada no meio acadêmico, já existem várias abordagens sobre o tema. [Krotov and Silva 2018] apresenta os limites legais do Web Scraping, indicando quais dados podem ser coletados e como a coleta deve ser feita dentro da lei. [Diouf et al. 2019] traz um panorama do estado da arte no quesito do Web Scraping, apresentando algumas ferramentas que agilizam o processo. Vargiu [Vargiu and Urru 2012] aplica os princípios do Web Scraping para montar um sistema de recomendações colaborativo. Gorro [Gorro et al. 2018] apresenta um esboço de sistema para identificação e classificação de Cyberbullying no Facebook usando Selenium para realizar a extração de dados. Galdino [Galdino et al. 2020] usa o Web Scraping para propor uma automatização da alimentação de dados públicos no Portal dos Números, que é feita manualmente. Também tratando de dados públicos, Furtuoso [dos Santos and D'Emery 2022] traz um sistema que cruza dados extraídos dos portais da transparência através de web scraping com dados do Auxílio Emergencial para detectar inconsistências nos pagamentos do auxílio. Porém, com a constante mudança na Internet, se faz necessária uma atualização sobre as dificuldades de se realizar Web Scraping, juntamente de suas soluções.

O objetivo principal deste estudo é elucidar e solucionar os diversos obstáculos enfrentados pelo cientista no processo de extração de dados da Internet. No presente caso de estudo, obtém-se avaliações de turistas no website Tripadvisor, juntamente de dados complementares, como uma nota de 0 a 5 dada pelo turista, data e local de origem do

autor da avaliação. São avaliadas duas versões para se obter esses dados, uma estática e outra dinâmica, expondo suas vantagens, problemas e soluções.

Na Seção 2 são listados alguns trabalhos relevantes envolvendo Web Scraping. Na Seção 3, é elucidada a maneira pela qual se decorreu a pesquisa, enquanto que na Seção 3.1 é exposto de forma geral o problema do Web Scraping. Na Seção 3.2 são elucidadas as diferentes maneiras de se percorrer o website, baseadas na maneira como os dados estão dispostos, e na Seção 4 são apresentados os resultados obtidos após os testes feitos, bem como problemas encontrados e soluções criadas para esses problemas. Por fim, a Seção 5 dispõe de aspectos gerais notados ao fim da pesquisa.

2. Fundamentação e trabalhos relacionados

Para se aprofundar no tema de Web Scraping, é necessário ter conhecimento de alguns conceitos que permeiam o tema: como são feitos os websites, e o do que se trata o Web Scraping. Assim, nas próximas seções serão apresentados esses conceitos, caracterizando-os com o estudo de caso.

2.1. Websites

A disposição de conteúdos online é feita, majoritariamente, por HTML - Hypertext Markup Language, ou Linguagem de Marcação de Hipertexto. O HTML é uma linguagem de marcação usada para transportar documentos pela internet, independente da plataforma [Berners-Lee and Connolly 1995]. Sua estrutura é composta por tags que exercem um papel pré definido na hora da renderização (disposição dos elementos textuais na formatação desejada) da página. Uma Tag define um elemento de um documento HTML, que possui dados a serem mostrados e regras de renderização atreladas.

Um documento HTML possui uma estrutura básica cabeçalho/corpo. O cabeçalho indica à ferramenta de renderização quais são os metadados da página: Títulos, fontes de conteúdo para agregar na renderização, links para scripts e fontes de caracteres, etc. O corpo, por sua vez, é a estrutura propriamente dita. O Quadro 1 apresenta um exemplo de HTML que dispõe elementos fundamentais na estruturação de um documento HTML, como títulos, listas de itens e textos em negrito.

Quadro 1. Exemplo de código HTML (baseado em RFC, 1995)

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
    <HTML>
    <!-- Este e um bom lugar para colocar comentarios-->
    <HEAD>
    <TITLE>Exemplo estrutural</TITLE>
    </HEAD><BODY>
    <H1>Primeiro titulo</H1>
    <OL>
    <LI>Primeiro item numa lista ordenada.
    <LI>Segundo item numa lista ordenada.
      <UL COMPACT>
      <LI> Note que listas podem ser aninhadas
      <LI> Espacos podem ajudar a ler
             o
```

```
            Codigo fonte HTML.
          </UL>
        </OL><P>
        Certifique-se de ler estas <b>Instrucoes importantes
        </b>.</P>
    </BODY></HTML>
```

Os elementos no HTML podem ser aninhados um dentro do outro, dando à estrutura a hierarquia de conteúdos que mais faz sentido com o que está sendo apresentado. Por exemplo, no Quadro 1, o texto "Note que listas podem ser aninhadas" está dentro da seguinte hierarquia de tags: `HTML -> BODY -> OL -> LI -> UL -> LI`.

Portanto, para acessar tal texto, deve-se percorrer toda essa hierarquia de tags. Porém, HTML puro não atende completamente aos anseios do mercado. A criação de websites complexos, com animações e outros tipos de enfeites que atraem os usuários e facilitam a leitura do website é comumente acompanhada de arquivos CSS - Cascading Stylesheets, ou Folhas de estilo em cascata. O objetivo do CSS é adicionar estilo, animação ou posicionamento aos elementos de um documento HTML [Firefox 2020, Atkins et al. 2022].

Para aplicar um arquivo de CSS em um HTML, basta, dentro das tags HEAD do documento HTML, importar o CSS. Dentro do arquivo CSS, criam-se vários identificadores, cada um contendo características que serão incorporadas pelas tags que receberem estes identificadores. A importância deste conhecimento reside no uso destes identificadores no software de scraping para identificar e extrair o conteúdo desejado.

Além do HTML e CSS, JavaScript é considerada a língua franca do ecossistema web, permitindo incorporar no documento o dinamismo da programação [Horváth and Menyhárt 2014]. JavaScript é uma das linguagens mais difundidas entre os programadores, de acordo com a última versão da pesquisa anual do StackOverflow[1]. Manipulação de strings, condições, repetições e outros recursos do JavaScript são vitais para acrescentar vida ao documento HTML. Devido a essa flexibilidade, essa linguagem vem ganhando diversos frameworks de desenvolvimento, como: Vue.JS, React, Angular, Svelte e Flight.

De acordo com [Li and Broadwater 2004], um framework é um conjunto de funcionalidades que já vêm prontas para uso, e que são comuns a várias aplicações, permitindo ao desenvolvedor focar apenas nas principais funcionalidades da aplicação. Os frameworks evitam o retrabalho, dando ao programador mais tempo para focar nas principais funcionalidades da aplicação. No VueJS existe a possibilidade de dividir o website em pequenas seções, e reutilizá-las caso seja necessário. Isso permite encontrar padrões no código HTML gerado pelo framework e, em cima desses padrões, construir a lógica da extração dos dados.

Outro ponto importante de como os websites são gerados com frameworks JavaScript é a capacidade de se iterar em cima de listas de dados. É comum no desenvolvimento web receber uma lista de itens advindos do banco de dados e, utilizando uma estrutura de repetição, mostrar cada um dos elementos dessa lista na tela. Neste caso, a estrutura

[1] https://survey.stackoverflow.co/2022/#technology-most-popular-technologies

mais provável de se encontrar em websites mais modernos seria um estrutura com muita repetição de classes e de aninhamentos de tags.

Uma observação importante é de que nem sempre as tags aparecem na mesma ordem. O algoritmo que as gera dinamicamente pode, por razões de estado do computador, renderizar na ordem invertida, ou então trocar alguns elementos. Isso afeta diretamente a atividade de scraping, sendo desejável usar o mínimo possível de posicionamento de tags como referência ao conteúdo desejado.

2.2. Web Scraping

O Web Scraping é um software que permite a extração e organização de dados com a simulação da interação humana em websites da Internet [Khder 2021]. Por sua ampla disseminação, várias bibliotecas foram criadas em diversas linguagens, na tentativa de oferecer uma maior gama de ferramentas para a atividade.

De nome similar, também existe o Web Crawling, que pode muitas vezes confundir o pesquisador que ainda não explorou a área de extração de dados. É importante frisar a diferença entre essas duas técnicas. Enquanto o Scraping visa a extração e formatação dos dados, o Crawling tem como objetivo indexar as páginas para uso posterior, como popular bancos de dados de motores de busca (p.ex. Google, Bing e DuckDuckGo) [Olston and Najork 2010].

A prática do Web Scraping, caso não seja feita com cuidado, pode acarretar em penalidades jurídicas por extrair dados de forma ilegal ou prejudicar a entidade detentora dos dados. Krotov [Krotov and Silva 2018] cita algumas situações onde realizar scraping pode acarretar em penalidades legais. Inicialmente, o scraping pode ser proibido pelo responsável do website nos termos de uso, que devem ser aceitos logo ao entrar na página. Outras situações onde o scraping é proibido, de acordo com o autor, são:

- Caso explore vulnerabilidades do website para se apoderar de informações privilegiadas;
- Caso prejudique o bom funcionamento do website; e
- Caso utilize dados de terceiros que visitaram o website, de forma a identificá-los.

Levando em conta essas observações, o web scraping pode ser empregado para a obtenção de dados da Internet ao se explorar as estruturas HTML e CSS dos websites. Com a exploração do conteúdo em JavaScript, pode-se ainda aprimorar a obtenção dos dados do website, já que esses dados são gerados dinamicamente.

3. Metodologia

Esta pesquisa se vale do método exploratório, realizando experimentos em ambientes controlados, e é desenvolvida como um estudo de caso. De acordo com Soares [Soares et al. 2018], um estudo de caso envolve analisar vários aspectos do objeto de pesquisa, que neste caso de estudo se trata de como implementar um scrapper para extrair dados do website Tripadvisor de forma automatizada. São implementadas duas abordagens de extração de dados: uma estática e outra dinâmica. Em cada implementação, são realizadas algumas extrações de dados, sendo avaliados os gastos em recursos computacionais e a velocidade na execução da tarefa. Também é avaliado o impacto da atividade de extração nos mecanismos de segurança do website.

3.1. O Algoritmo Genérico

A abordagem geral para extração de dados na internet pode ser descrita em uma simples sentença: "Ao entrar na página alvo, baixe e salve em um arquivo os dados importantes". Embora seja simples, esta instrução pode acarretar em diversos problemas, que dependem do objetivo e da natureza do objeto alvo do scraping. Se, hipoteticamente, o objetivo do scraping é recolher a lista de cores presentes na página "Lista de Cores" do Wikipédia[2], todo o conteúdo está concentrado em apenas uma página, e nada mais (um exemplo deste scraping pode ser encontrado no Github do autor [3]). Dessa forma, o script não precisa se preocupar com a navegação entre diferentes páginas.

Quando, porém, os dados estão distribuídos em várias páginas subsequentes, a sentença descrita anteriormente muda: "para cada página existente, extraia os seus dados e avance de página, até não existir próxima página". Esta lógica pode ser considerada o algoritmo genérico para extração de dados distribuídos sequencialmente em diversas páginas. O Algoritmo 1 ilustra em pseudo-código das ações citadas na forma de um algoritmo.

Algoritmo 1: Obter dados de várias páginas

```
   Saída: dados finais
1  dados_finais = {};
2  repita
3  |  dados_finais = dados_finais + extrai_dados_da_pagina_atual();
4  |  passa_de_página();
5  até todas as páginas foram extraídas;
```

Porém, em nenhum momento o algoritmo genérico mostra como deve ser feita a navegação entre páginas, indo para uma próxima página, e como ele identifica que chegou ao fim do conteúdo. O motivo dessa ausência é justificável, pois depende de qual abordagem é usada no algoritmo final, a estática ou a dinâmica, que são apresentadas a seguir.

3.2. As abordagens estática e dinâmica

A separação entre diferentes tipos de abordagens pode ser feita levando em conta a maneira pela qual o extrator de dados terá acesso ao conteúdo do website alvo. A primeira abordagem será chamada de estática, pois faz o download da página no momento em que ela é acessada pelo extrator. Isso significa que nenhum conteúdo gerado a partir da interação do usuário, como cliques ou preenchimento de campos, será baixado e lido pelo extrator.

Já a segunda abordagem será chamada de dinâmica, pois visa não só acessar a página como dar ao usuário a possibilidade de interagir com ela. Diferentemente da estática, a dinâmica simula um navegador de internet em tempo real, permitindo que

[2]https://pt.wikipedia.org/wiki/Lista_de_cores

[3]https://github.com/JoaoPedroMDP/colorScrapper

o extrator simule as interações do usuário com o website, como cliques em botões e preenchimento de formulários.

Nesta pesquisa, as duas abordagens foram implementadas, sendo que cada implementação é considerada uma versão diferente: uma versão estática, apresentada no Algoritmo 2, e uma dinâmica, apresentada no Algoritmo 3.

Algoritmo 2: Extração de dados com abordagem estática.

```
Entrada: link_inicial (Link inicial do website)
Saída: Dados importantes coletados
1  URL = link_inicial;
2  dados_importantes = {};
3  enquanto URL faça
4      página = obter_html(URL);
5      dados_importantes = interpretar_html(página);
6      URL = gerar_nova_url(URL);
7  fim
8  retorna dados_importantes;
```

Uma descrição breve do Algoritmo 2 pode ser: ”Enquanto tiver uma URL válida, acesse a URL, baixe a página, colete todos os dados importantes, gere uma nova URL e repita os passos anteriores, até não existir mais uma URL válida. Em seguida, retorne todos os dados coletados”.

Algoritmo 3: Extração de dados com abordagem dinâmica.

```
Entrada: link_inicial (Link inicial do website)
Saída: Dados importantes coletados
1  dados_importantes = {};
2  navegador = abrir_navegador(link_inicial);
3  repita
4      remove_obstaculos(navegador);
5      dados_importantes = dados_importantes + extrair_dados(navegador);
6      mecanismo_avanço = retornar_botao_avanco(navegador);
7      se mecanismo_avanço != null então
8          mecanismo_avanco.clicar();
9      fim
10 até mecanismo_avanço == null;
11 retorna dados_importantes;
```

Já no Algoritmo 3, a descrição breve seria: ”Acesse a página inicial pelo navegador simulado e repita os pontos a seguir. Remova obstáculos, como cookies e propagandas; extraia os dados importantes; e se encontrar algum mecanismo de avanço, clique nele. Quando não existir mais mecanismo de avanço, retorne todos os dados coletados.”

4. Resultados

Com os algoritmos implementados, realizou-se a extração de dados de 10 pontos turísticos diferentes. O website utilizado para se extraírem os dados foi o Tripadvisor[4], por ser uma plataforma de turismo reconhecida e amplamente utilizada [Caracristi et al. 2021a, Feger et al. 2023, Feger et al. 2024, Kaizer et al. 2021, Gosenheimer et al. 2021, Caracristi et al. 2021b]. Foram observados alguns comportamentos interessantes, como estouros de memória e bloqueios do IP por parte do Tripadvisor. As análises feitas estão detalhadas a seguir.

4.1. Navegação entre páginas

A versão estática do extrator se vale de alterações na URL para indicar qual página de comentários deseja ver a seguir. A cada página avançada, a URL é alterada indicando um deslocamento de comentários. Isso significa que, caso o deslocamento seja de 10 comentários, a página vai mostrar do comentário 11 em diante. Como cada página mostra 10 comentários. Então, a lógica que rege o identificador de páginas na URL obedece a seguinte função: $(paginaAtual - 1) * 10$. Na segunda página, o identificador seria, p.ex., `https://www...Reviews-or10-Jardim_Botanico...html`, o que significa que os 10 primeiros comentários foram ignorados e são mostrados os 10 seguintes. Na terceira página a URL seria `https://www...Reviews-or20-Jardim_Botanico...html`, e assim por diante.

Já na versão dinâmica, o avanço de páginas é feito clicando em botões de navegação que o website disponibiliza ao usuário. É comum que em situações onde uma série de dados organizados estão distribuídos em diversas páginas existam botões de navegação, feitos para facilitar a visualização do usuário. Assim a implementação dinâmica precisa simular um usuário e clicar nos botões para avançar a página.

4.2. Coleta de informações

Para coletar os dados, ambas as versões precisam percorrer o documento HTML e interpretar a estrutura descrita nele. Inicialmente, a versão estática usa o **id** de cada tag que contêm o dado necessário. O id é um atributo que pode ser adicionado em uma tag HTML para identificá-la de forma única[Raggett et al. 2018]. Isso significa que o id deve ser único, ao contrário das classes, que podem se repetir em várias tags diferentes, como apresentado no Quadro 2.

Quadro 2. Exemplo de id em uma tag HTML

```
<body>
    <div id="um-id">
        <input id="outro-id">
        </input>
    </div>
</body>
```

[4]https://www.tripadvisor.com.br/

Porém, após a Tripadvisor atualizar o website, quase todos os id's foram removidos, quebrando por completo a lógica de extração. Depois dessa atualização, por exemplo, a tag que contém o título da avaliação passou a ser algo como: `<span class="yCeTE">Fotos lindas!</span>`. Note que não existe nenhum id para identificar essa tag específica.

Assim, torna-se importante usar um método de localização de conteúdo que não dependa inteiramente dos ids. Para essa tarefa, pode-se usar XPath. O XPath é o caminho de tags a ser percorrido até chegar no conteúdo desejado, obedecendo a hierarquia do documento [Robie et al. 2017]. Se, no exemplo do Quadro 3, é desejado extrair o conteúdo "Target data", pode-se usar o seguinte XPath: `./body/div[0]/p[1]`. Desse modo, usa-se a própria estrutura do documento HTML para alcançar a informação desejada.

Quadro 3. Exemplo de arquivo HTML

```
<body>
    <div class="content">
        <p class="header-data"> Some useless data</p>
        <p class="data">Target data</p>
    </div>
    <div class="content">
        <p class="data">More useless data</p>
    </div>
</body>
```

Porém, os websites frequentemente mudam suas estruturas ao longo do tempo, para se adaptarem a novas tendências de mercado ou para otimizar a experiência do usuário. Juntamente com essas mudanças, os scripts de scraping precisam ser atualizados, já que dependem da estrutura da página para localizar o dado alvo. Com isso, fica evidente que quanto menos o algoritmo depender de padrões estruturais da página, mais o scrapper é resistente às atualizações do website.

Para não depender tanto da estrutura da página, é possível melhorar o uso do XPath, incluindo outros elementos da página para a localização do conteúdo. Neste caso, são usadas as classes CSS. No mesmo exemplo (Quadro 3), é possível extrair "Target data" especificando no XPath a classe "data" da seguinte maneira: `//div[0]/p[contains(@class, "data")]`. Deste modo, se a classe permanece a mesma, os autores da página podem trocar a tag de lugar que o dado correto ainda é extraído.

Dados os algoritmos genéricos listados nas Subseções 3.1 e 3.2, e embora as duas abordagens possuam suas vantagens e desvantagens, a abordagem dinâmica possui algumas características que a tornam a "necessária" no caso de obter as avaliações do website Tripadvisor, pois:

- Permite selecionar o idioma das avaliações;
- Permite passar para o próximo conjunto de avaliações sem implementar a lógica de URL da versão estática;

- Permite remover obstáculos, como pop-ups para aceitar cookies ou propagandas que ocupam a tela; e
- Permite utilizar recursos e filtragem disponibilizados pelo website.

Para a implementação estática do algoritmo, utilizou-se a biblioteca BeautifulSoup4[5]. Ela possui um interpretador de XML (estrutura "pai" do HTML) embutido, bem como diversas funções que facilitam a navegação da página baixada.

Já na versão dinâmica foi usada a biblioteca Selenium[6], já bem conhecida no campo de automatizações de tarefas e testes de front-end, justamente por permitir que o usuário programe interações com a página. Assim como a BeautifulSoup4, o Selenium também possui um interpretador de XML embutido.

A versão estática, disponível em [7], embora tenha sido capaz de extrair os comentários necessários, não foi capaz de satisfazer a demanda completa. O primeiro empecilho foi de que os comentários eram coletados em inglês, visto ser impossível para o extrator estático interagir com o documento e selecionar o idioma. Depois, por depender da URL para navegar entre as várias páginas de comentários, possuía uma lógica mais complexa de avanço (passar para a próxima página), já que o identificador da página variava de 10 em 10 ou de 5 em 5, o que dificulta a manutenção do código.

No mais, essa versão estática atendeu parcialmente o processo de extração dos comentários, sendo capaz de gerar um documento estruturado em CSV com os comentários recolhidos.

Já a versão dinâmica, disponível no GitHub [8], se provou mais flexível, possibilitando escolher o idioma dos comentários. Para avançar as páginas, utilizou-se o próprio botão de paginação, como disposto no Quadro 4, e isso traz uma vantagem grande. Quando usamos o botão de paginação, e não a URL, mantemos em memória o idioma selecionado, evitando ter de selecionar de novo e esperar mais uma requisição de dados para o servidor do website.

Quadro 4. Trecho para o avanço de páginas

```
    def has_next_page(self):
        try:
            if self.driver.find_element(By.XPATH, XPATHS["
               next_page_button"]) is not None:
                return True
        except NoSuchElementException:
            debug("Acabaram-se as paginas!")
            return False
        except Exception as e:
            debug("Erro ao verificar se ha proxima pagina: {}".
               format(e))
            return False
```

[5]https://pypi.org/project/beautifulsoup4/
[6]https://pypi.org/project/selenium/
[7]https://drive.google.com/file/d/12T2nJl6r60MWg19ehR18d2DLIcrv84Ee/view?usp=sharing
[8]https://github.com/JoaoPedroMDP/review-scraping

Existe ainda uma outra vantagem de se abordar o problema pelo viés dinâmico envolvendo os filtros nativos do website. Quando avança para a seção de comentários no website, existe um botão intitulado “Filtros”. Ali, é possível combinar categorias de filtros diferentes, como: Avaliação em estrelas, mês de escrita do comentário e o tipo da visita. Embora essa vantagem não tenha sido explorada neste trabalho, é totalmente viável e permite ao extrator ter maior controle sobre qual tipo de conteúdo ele deseja extrair do ponto turístico.

Dadas as características apresentadas, é possível estabelecer uma relação de qual abordagem tem um desempenho maior, baseada na tarefa exercida. A abordagem dinâmica tem um melhor desempenho quanto à abrangência de conteúdo, em termos de localização na página, e também quando exige-se um certo nível de flexibilidade na hora de percorrer a página alvo. Já a abordagem estática desempenha melhor quando os recursos computacionais são limitados, ou quando é necessária uma alta velocidade do extrator - já que a página é processada sem a necessidade de se simular o navegador de Internet.

4.3. Melhorando a eficiência

Algumas características da implementação precisam ser levadas em consideração para otimizar a execução do extrator. Esses itens são apresentados em mais detalhes a seguir.

4.3.1. Gerenciamento de memória

Devido à grande quantidade de comentários que alguns pontos turísticos podem ter, é importante se atentar para o limite de memória principal. A quantidade de memória principal disponível pode ser um fator limitante quando o ponto turístico possui uma quantidade de avaliações bem acima da média. Para solucionar esse problema, ao invés de salvar todos os comentários de uma só vez (quando a última página for extraída), pode-se salvar os comentários periodicamente no arquivo final. Dessa maneira, é possível limpar a memória principal para guardar os próximos comentários. E essa solução ainda auxilia em outro quesito: caso o computador, por algum motivo, desligue, ou o programa seja interrompido previamente, não se perde todo o trabalho feito, já que de tempos em tempos os comentários são consolidados em memória persistente.

Outro ponto concernente à memória principal é quanto à renderização de uma página web. É um processo que gasta memória e tempo de processamento, diminuindo a velocidade da extração. Para amenizar o impacto do extrator nos recursos do computador, o código pode ser adaptado para não renderizar a janela do navegador, realizando as operações diretamente na estrutura de dados da página web. Essa configuração pode ser feita ao usar a opção “–headless” do driver de navegador – no presente caso, o Google Chrome.

4.3.2. Negação de serviço

É importante também se atentar para os mecanismos de segurança que os websites utilizam. Dependendo da densidade de requisições que o scrapper fizer (quantidade de

requisições por tempo), o website pode bloquear o computador por interpretar como uma atividade maliciosa.

Após um estudo mais a fundo sobre os mecanismos de segurança que os websites geralmente usam, chegou-se à conclusão de que a atividade do scrapper pode comumente ser interpretada como um ataque Negação de Serviço (Denial of Service - DoS, em inglês). Zargar, Josh e Tipper [Zargar et al. 2013] segregam os ataques DoS em 2 grandes categorias: ataques de Infraestrutura e ataques de Aplicação. A primeira explora mais a fundo a infraestrutura de rede, se valendo de brechas em protocolos da camada de Transporte do modelo OSI, ou de camadas inferiores. A segunda envolve a camada de aplicação. Esses ataques visam mais a infraestrutura total da aplicação, como seu consumo de CPU e RAM. Ataques desta categoria são categorizados como "de aplicação" pois se utilizam de brechas na própria aplicação atacada, como protocolos de camada 7 do modelo OSI ou regras de negócio do software.

Dentro dos ataques de aplicação existem várias subcategorias. A que mais se aproxima do provável motivo do bloqueio é a de Ataques de Inundação de HTTP (*HTTP Flooding Attacks*). Esta se divide ainda em subtipos. O primeiro é o de Inundação de Sessões, onde o atacante tenta estabelecer várias sessões com o alvo. Porém o extrator implementado não se encaixa neste tipo de ataque pois a mesma sessão é utilizada do começo ao fim. Isso fica evidente pois só é necessário aceitar *cookies* e termos de uso na primeira requisição feita pela biblioteca. Mas é importante notar que é utilizada uma sessão por *thread*. Então, se muitas *threads* forem abertas, pode levar o servidor a caracterizar como um ataque deste tipo. No caso de teste, essa hipótese foi minimizada pois foram usadas apenas 5 *threads*.

Quadro 5. Seção do arquivo de log gerado pela aplicação

```
[2022-10-03 15:23:22,294] DEBUG - Tangua_Park -> Indo para proxima
   pagina
[2022-10-03 15:23:22,825] DEBUG - Jardim_Botanico_de_Curitiba ->
   Indo para proxima pagina
[2022-10-03 15:23:23,086] DEBUG - Estrada_Da_Graciosa -> Indo para
   proxima pagina
[2022-10-03 15:23:24,130] DEBUG - Museu_Oscar_Niemeyer -> Indo para
    proxima pagina
[2022-10-03 15:23:24,470] DEBUG - Parque_Barigui -> Indo para
   proxima pagina
[2022-10-03 15:23:31,791] DEBUG - Tangua_Park -> Indo para proxima
   pagina
[2022-10-03 15:23:31,907] DEBUG - Jardim_Botanico_de_Curitiba ->
   Indo para proxima pagina
[2022-10-03 15:23:32,060] DEBUG - Estrada_Da_Graciosa -> Indo para
   proxima pagina
[2022-10-03 15:23:32,752] DEBUG - Museu_Oscar_Niemeyer -> Indo para
    proxima pagina
[2022-10-03 15:23:33,390] DEBUG - Parque_Barigui -> Indo para
   proxima pagina
[2022-10-03 15:23:38,974] DEBUG - Tangua_Park -> Indo para proxima
   pagina
[2022-10-03 15:23:39,254] DEBUG - Estrada_Da_Graciosa -> Indo para
   proxima pagina
```

```
[2022-10-03 15:23:39,356] DEBUG - Jardim_Botanico_de_Curitiba ->
   Indo para proxima pagina
[2022-10-03 15:23:39,934] DEBUG - Museu_Oscar_Niemeyer -> Indo para
    proxima pagina
[2022-10-03 15:23:41,144] DEBUG - Parque_Barigui -> Indo para
   proxima pagina
[2022-10-03 15:23:47,015] DEBUG - Estrada_Da_Graciosa -> Indo para
   proxima pagina
```

Também foi estudado o ataque de Inundação de Requisições, onde o atacante inunda o alvo com várias requisições, usualmente 10 ou mais por segundo. Ao analisar os logs do extrator implementado, este não chega nem a 10% dessa taxa. De acordo com as linhas dispostas no Quadro 5, é possível observar que são feitas 16 requisições em 25 segundos, aproximadamente, o que nos dá uma taxa de menos de uma requisição por segundo.

Existe ainda o Ataque de Tempo de Resposta Demorado. Este ataque se vale de fazer requisições para funcionalidades da aplicação que realizam um alto processamento, exigindo muitos recursos do computador. No caso estudado, o IP do computador de teste chegou a ser bloqueado pelo Tripadvisor, e, levando em consideração as categorias apresentadas, o ataque que mais chega perto de ser a provável causa do bloqueio é o de Tempo de Resposta Demorado.

Observando as requisições nas ferramentas de desenvolvedor, foi possível observar que quando o usuário avança a lista de comentários, uma requisição que consome muitos recursos é ativada. Na Tabela 1 é apresentada uma lista sumarizada contendo o tamanho e tempo de resposta para todas as requisições que trazem os próximos 10 comentários. O Tamanho mínimo é 19 KB e o máximo 21 KB. Já o tempo de resposta mínimo é 471 ms e o máximo 730 ms.

Tamanho (KB)	Tempo de resposta (ms)
19	730
19,5	531
19,6	575
18,5	571
19	471
19	596
19	588
20	560
21	547

Tabela 1. Lista de tempos e tamanhos de requisições de comentários.

Na Tabela 2 são apresentados os tamanhos e tempos de requisições esporádicas do website, provavelmente para coletar estatísticas de uso e etc. O tamanho mínimo é 0,162 KB e o máximo 0,518 KB. O tempo mínimo é 157 ms e o máximo 181. Vê-se então que a diferença é de até 100 vezes menos quantidade de dados e 4 vezes menos tempo.

Tamanho (KB)	Tempo de resposta (ms)
0,162	166
0,296	173
0,518	157
0,493	158
0,358	181

Tabela 2. Lista de tempos e tamanhos de requisições esporádicas.

4.3.3. NAT - Network Address Resolution

NAT (*Network Address Translation*, ou Tradução de Endereços de Rede) é um técnica que permite que um conjunto de endereços de Internet (IP) se comuniquem com outro conjunto de endereços IP através de um único IP para cada conjunto [Srisuresh and Holdrege 1999]. Observe na Figura 1 que a conexão de todos os computadores com a Internet é intermediada pelo endereço IP 11.0.0.2. Todos os computadores possuem IP iniciado com 10.0.0.* e ”saem”para a Internet utilizando o IP 11.0.0.2.

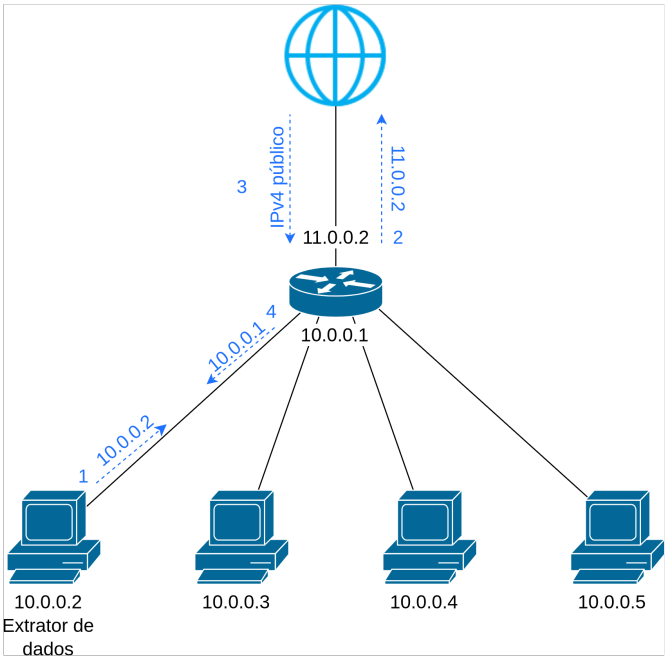

Figura 1. Disposição de uma rede sob NAT.

Ao trazer esta disposição de rede para a situação do extrator, identifica-se que, ao executar o scrapper a partir de um único computador, e o website alvo bloquear o IP de origem (11.0.0.2), acabará bloqueando os outros computadores da rede local. Isso significa que não só o computador que executa o extrator será bloqueado, mas sim todos os outros e até outras redes que estiverem debaixo do mesmo IP público também serão

bloqueadas. Desse modo, é de responsabilidade do desenvolvedor se atentar a este fato para não provocar problemas para si e também para outros usuários.

4.3.4. Aplicando Multithreading

Para aumentar a capacidade de extração de um scrapper, pode-se fazer o emprego de *Multithreading*. *Multithreading* é um técnica de programação que permite que um único processo alterne entre as tarefas definidas pelo programador, aumentando a *performance* do processador [Saavedra-Barrera et al. 1990].

O Quadro 6 apresenta um exemplo de aplicação de Multithreading para web scraping. Inicia com o número de threads desejado e uma lista com todas as URLs a serem analisadas. O algoritmo irá iniciar o número indicado de threads, e cada uma irá remover da pilha de URLs um ponto turístico para ser analisado. Assim que uma thread termina sua execução, o scrapper inicia uma nova thread com uma nova URL, e assim por diante - sempre respeitando o máximo de threads definido em configuração.

Quadro 6. Exemplo de aplicação de Multithreading no scraping

```
NUM_THREADS = 3
lista_de_urls = ["url1", "url2"]

def url_task(URL: str, directory: str):
    """Aqui dentro fica a logica do scraping"""
    pass

with futures.ThreadPoolExecutor(NUM_THREADS) as executor:
    future_results = {
        URL: executor.submit(url_task, URL, begin_time)
            for URL in lista_de_urls
    }

    for URL, future in future_results.items():
        try:
            future.result()
        except Exception as exc:
            print("Thread "+URL+" gerou uma excecao:")
            traceback.print_exception(type(exc), exc, exc.
               __traceback__)
```

No extrator dinâmico, antes da otimização, a velocidade da extração era de aproximadamente 142 comentários por minuto. Com 5 threads simultâneas, obteve-se uma velocidade de 457 comentários por minuto, ou seja, uma melhora de 221% com relação à execução com apenas uma thread. É importante frisar que essa melhora não acelera a velocidade com que uma URL é analisada, evitando que a extração seja identificada como maliciosa, mas sim avaliando o montante total de trabalho a ser realizado considerando

os vários destinos (websites) a serem analisados.

4.3.5. Lista de IP Malicioso

Ainda tem-se que considerar a existência das listas de IPs maliciosos, ou RBLs (*Realtime Blackhole List*). Essas listas são mantidas por servidores e são utilizadas por empresas para bloquear IPs e evitar ataques.

O website WhatIsMyIp[9] é um exemplo de servidor que, dentre suas funcionalidades, existe a possibilidade de consultar a reputação de um IP. Ao inserir o IP de um computador, o website retorna se o IP foi listado como malicioso em diversos RBLs (Realtime Blackhole List) - servidores que proveem listas de IPs potencialmente maliciosos.

Um exemplo de como se daria esse fluxo de bloqueio pode ser como o seguinte:

- O script executa várias requisições em um website;
- A segurança do website entende que é uma atividade maliciosa;
- Além de bloquear o IP do computador, envia para os provedores RBL este IP;
- Provedores RBL adicionam o IP em suas listas, disponibilizando-as para outros clientes.

Um dos provedores de blackhole acessados na pesquisa foi o SpamRats [10]. Neste provedor é possível requisitar a remoção do IP da lista, mas não se sabe ao certo quanto tempo demora para que essa retirada seja refletida na rede global. É importante mencionar também que existem vários RBLs espalhados pela Internet, e requisitar a retirada de um IP não é uma funcionalidade presente em todos eles.

No nosso caso, provavelmente o IP fornecido pela operadora para o computador do scrapper foi listado em um desses provedores. Depois de pedirmos a remoção do IP da lista de IPs maliciosos e se passar aproximadamente uma semana, já foi possível acessar os websites normalmente, não sendo possível identificar diretamente a causa da suspensão do bloqueio. Esta suspensão, após um tempo relativamente curto, levanta três hipóteses, não testadas:

- O pedido de suspensão do bloqueio foi atendido;
- O primeiro bloqueio é temporário, servindo de aviso ao IP de que seu comportamento não é bem-vindo na Internet; e
- O scrapper não possui a capacidade de inserir na rede grande quantidades de dados a ponto de perturbar o tráfego de outros dispositivos e, portanto, uma punição menor foi aplicada ao IP.

A comprovação dessas hipóteses - fora do escopo deste estudo - poderia ser feita ou conhecendo as regras que os RBLs usam para aplicar as punições, ou realizando testes de diversas magnitudes até ser possível encontrar um padrão na punição.

[9] https://www.whatismyip.com/
[10] www.spamrats.com

5. Conclusão

Numa Internet voltada para o consumo humano, a extração automatizada encontra diversas barreiras, pois os sites buscam agradar e atrair a atenção dos leitores e potenciais consumidores, ofertando algo novo em intervalos cada vez menores. Visando minimizar o impacto dessas barreiras, foi explicado sobre as tecnologias usadas na construção de websites, bem como mecanismos de se localizar os dados desejados ao se criar um extrator automatizado. O estudo de caso envolveu implementar um scrapper para extrair dados do website Tripadvisor de forma automatizada.

Um algoritmo geral foi apresentado e as abordagens de implementação estática e dinâmica foram avaliadas. Os obstáculos que não poderiam ser vencidos com a extração estática foram contornados com a extração dinâmica. Assim, todos os dados necessários podem ser coletados e os problemas que surgiram ao longo dos testes foram sanados, e suas soluções documentadas. Foram apresentados os limites das otimizações, especialmente na seção sobre Negação de Serviço (DoS), para não incorrer em bloqueios indesejados por parte do provedor do website. Foi possível também aumentar a capacidade do extrator em até 221% quando configurado para extrair dados de 5 pontos turísticos diferentes, utilizando programação concorrente (multithreading).

O extrator apresentado possui a limitação de estar suscetível a mudanças na página alvo, o que implica na necessidade de se atualizar as classes e os XPaths usados para identificar os elementos. Uma evolução do extrator seria alterar a lógica de extração e adicionar modelos de linguagem natural para realizar a identificação do conteúdo alvo da extração. Essa melhora não só eliminaria a necessidade de se modificar o extrator a cada atualização na página alvo, como removeria a necessidade dos cientistas conhecerem sobre a estrutura da página alvo para realizarem a extração, democratizando ainda mais o acesso à informação nas pesquisas científicas.

Referências

[Atkins et al. 2022] Atkins, T., Etemad, E. J., and Rivoal, F. (2022). Css snapshot 2022. https://www.w3.org/TR/css-2022/. Accessed at: 11/09/2023.

[Berners-Lee and Connolly 1995] Berners-Lee, T. and Connolly, D. (1995). Hypertext markup language - 2.0. https://www.rfc-editor.org/rfc/rfc1866. Acessado em: 21/03/2023.

[Caracristi et al. 2021a] Caracristi, M. F. A., Feger, J. E., da Silva, T. M., and Marynowski, J. E. (2021a). Uma viagem pelo jalapão, brasil: análise das experiências turísticas. *Revista Paranaense de Desenvolvimento (RPD)*, 41:89–110.

[Caracristi et al. 2021b] Caracristi, M. F. A., Feger, J. E., Marynowski, J. E., and Minasi, S. M. (2021b). A demanda turística do parque estadual do jalapão (pej, to, brasil) baseada em comentários de redes sociais. *Revista Brasileira de Ecoturismo*, 14:291–314.

[Diouf et al. 2019] Diouf, R., Sarr, E. N., Sall, O., Birregah, B., Bousso, M., and Mbaye, S. N. (2019). Web scraping: State-of-the-art and areas of application. Proc. of the 2019 IEEE International Conference on Big Data (Big Data). volume 17, pages 6040–6042. IEEE. doi: 10.1109/BigData47090.2019.9005594.

[dos Santos and D'Emery 2022] dos Santos, A. J. F. and D'Emery, R. A. (2022). Desenvolvimento de uma plataforma de dados abertos para análise do auxílio emergencial do brasil durante a covid-19: uma abordagem web scraping. Proc. do XVIII Simpósio Brasileiro de Sistemas de Informação (SBSI Estendido 2022). pages 197–206. Sociedade Brasileira de Computação (SBC).

[Feger et al. 2023] Feger, J. E., Marynowski, J. E., Botta, E. D., and de Fátima de Albuquerque Caracristi, M. (2023). Experiência vivenciada por turistas no fervedouro do ceiça, tocantins, to, brasil. *Revista Paranaense de Desenvolvimento*, 44:275–298.

[Feger et al. 2024] Feger, J. E., Marynowski, J. E., Reck, S. B., di Fátima Rocha Garcia, R., and de Fátima de Albuquerque Caracristi, M. (2024). Experiência turística no atrativo serra do espírito santo situado no parque estadual do jalapão (to). *Revista Brasileira de Ecoturismo (RBEcotur)*, 17:86–105. doi: 10.34024/rbecotur.2024.v17.14930.

[Firefox 2020] Firefox, M. (2020). Css: Cascading style sheets. https://developer.mozilla.org/en-US/docs/Web/CSS. Acessado em: 08/02/2023.

[Galdino et al. 2020] Galdino, I. M., Gallindo, E. D. L., and Moreira, M. W. L. (2020). Utilização de bots para obtenção automática de dados públicos usando as técnicas de web crawling e web scraping. Proc. of the WCGE - Workshop de Computação Aplicada em Governo Eletrônico. pages 172–179. Sociedade Brasileira da Computação. doi: 10.5753/wcge.2020.11269.

[Gorro et al. 2018] Gorro, K. D., Sabellano, M. J. G., Gorro, K., Maderazo, C., and Capao, K. (2018). Classification of cyberbullying in facebook using selenium and svm. Proc. of the ICCCS - Int. Conf. on Computer and Communication Systems. pages 183–186. IEEE. doi: 10.1109/CCOMS.2018.8463326.

[Gosenheimer et al. 2021] Gosenheimer, A., Feger, J. E., Minasi, S. M., Marynowski, J. E., and da Silva, T. M. (2021). Foz do iguaçu/pr na perspectiva da teoria do espaço turístico. *Marketing Tourism Review*, 6:1–28. doi: 10.29149/mtr.v6i2.6621.

[Horváth and Menyhárt 2014] Horváth, G. and Menyhárt, L. (2014). Teaching introductory programming with javascript in higher education. Proc. of the ICAC - Int. Conf. on Applied Informatics. pages 339–350. University of Debrecen/ Debreceni Egyetem.

[Kaizer et al. 2021] Kaizer, E. F., Caracristi, M. F. A., Feger, J. E., Marynowski, J. E., and Silva, T. M. (2021). Análise da experiência relatada pelos turistas ao visitar o parque estadual do jalapão (pej) – to, brasil. *Ateliê do Turismo*, 5:183–204.

[Khder 2021] Khder, M. (2021). Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and its Applications*, 13:145–168. doi: 10.15849/IJASCA.211128.11.

[Krotov and Silva 2018] Krotov, V. and Silva, L. (2018). Legality and ethics of web scraping. Proc. of the ERF - Emergent Research Forum.

[Li and Broadwater 2004] Li, F. and Broadwater, R. (2004). Software framework concepts for power distribution system analysis. *IEEE Transactions on Power Systems*, 19:948–956. doi: 10.1109/TPWRS.2003.821437.

[Oliveira et al. 2020] Oliveira, F. A. D., Villote, G. D. S., Costa, R. L., Goldschmidt, R. R., and Cavalcanti, M. C. (2020). Minerando regras de associação de multirrelação na web de dados. *iSys - Brazilian Journal of Information Systems*, 13:77–100.

[Olston and Najork 2010] Olston, C. and Najork, M. (2010). Web crawling. *Foundations and Trends® in Information Retrieval*, 4:175–246. doi: 10.1561/1500000017.

[Raggett et al. 2018] Raggett, D., Hors, A. L., and Jacobs, I. (2018). Html 4.0.1 specification. https://www.w3.org/TR/html401/.

[Robie et al. 2017] Robie, J., Dyck, M., and Spiegel, J. (2017). Xml path language (xpath) 3.1. https://www.w3.org/TR/xpath-31/.

[Saavedra-Barrera et al. 1990] Saavedra-Barrera, R., Culler, D., and von Eicken, T. (1990). Analysis of multithreaded architectures for parallel computing. Proc. of the second annual ACM symposium on Parallel algorithms and architectures - SPAA '90. pages 169–178. ACM Press. doi: 10.1145/97444.97683.

[Soares et al. 2018] Soares, A., Dorlivete, P., Shitsuka, M., Parreira, F. J., and Shitsuka, R. (2018). *Metodologia da Pesquisa Científica*.

[Srisuresh and Holdrege 1999] Srisuresh, P. and Holdrege, M. (1999). Ip network address translator (nat) terminology and considerations. https://www.rfc-editor.org/info/rfc2663.

[Vargiu and Urru 2012] Vargiu, E. and Urru, M. (2012). Exploiting web scraping in a collaborative filtering- based approach to web advertising. *Artificial Intelligence Research*, 2. doi: 10.5430/air.v2n1p44.

[Vieira and Moura 2020] Vieira, J. P. A. and Moura, R. S. (2020). Análise de métodos de extração de aspectos em opiniões regulares. *iSys - Brazilian Journal of Information Systems*, 13:82–97.

[Zargar et al. 2013] Zargar, S. T., Joshi, J., and Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys Tutorials*, 15:2046–2069. doi: 10.1109/SURV.2013.031413.00127.

[Zhao 2017] Zhao, B. (2017). *Web Scraping*, pages 1–3. Springer International Publishing. doi: 10.1007/978-3-319-32001-$4_483 - 1$.