

UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUSTAVO FABRO

**EXTRAÇÃO ESTRUTURADA DE DADOS EM FONTES HETEROGÊNEAS COM
WEB CRAWLERS**

CRICIÚMA

2018

GUSTAVO FABRO

**EXTRAÇÃO ESTRUTURADA DE DADOS EM FONTES HETEROGÊNEAS COM
WEB CRAWLERS**

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Gilberto Vieira da Silva

CRICIÚMA

2018

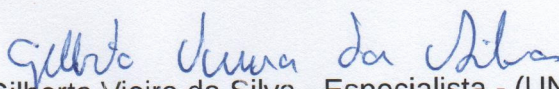
GUSTAVO FABRO

**EXTRAÇÃO ESTRUTURADA DE DADOS EM FONTES HETEROGÊNEAS COM
WEB CRAWLERS**

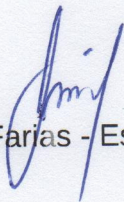
Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Desenvolvimento de Software para Web

Criciúma, 28 de junho de 2018.

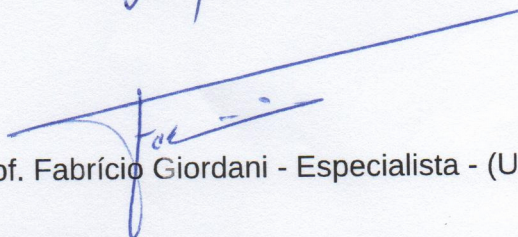
BANCA EXAMINADORA



Prof. Gilberto Vieira da Silva - Especialista - (UNESC) - Orientador



Prof. Anderson Rodrigo Farias - Especialista - (Faculdade SATC/UNIBAVE)



Prof. Fabrício Giordani - Especialista - (UNESC)

A Eru, imperador de Além-Mar.

AGRADECIMENTOS

No final desta etapa, quero deixar minha gratidão a todos aqueles que direta ou indiretamente me ajudaram, apoiaram e me incentivaram nesta caminhada acadêmica, que apesar de ser um pequeno passo, foi com toda certeza uma escolha decisiva, que de muitas formas ajudou e continuará ajudando na minha formação tanto profissional como pessoal.

A Deus, pelo dom da vida.

Agradeço principalmente ao meu pai e minha mãe que nunca mediram esforços para toda ajuda que precisei neste período.

A Maria Marlei Vargas (*in memoriam*).

A minha noiva Giulia Nascimento, por todo o carinho e compreensão nesta etapa.

Aos meus amigos, que também foram importantes em minha caminhada, em especial aos que cultivei durante o curso, e acompanharam de perto esta minha trajetória.

E por fim ao professor Gilberto, meu orientador, por aceitar este convite e ser uma peça importante na realização deste trabalho.

“Você é uma ótima pessoa Sr. Bolseiro, e gosto muito de você; mas, afinal de contas, você é apenas uma pessoazinha neste mundo enorme!” (Gandalf e Bilbo)

J. R. R. Tolkien

RESUMO

Com crescimento de dados na *web* torna-se cada vez maior a necessidade de ferramentas que auxiliam no consumo dessas informações. Dentre as categorias desses dados estão as fontes de notícias, em que há um grande número de portais disponíveis e no qual um determinado assunto pode ser tratado por diferentes sites. Com isso, o objetivo deste trabalho foi determinar formas de extração estruturada desses dados ao mesmo tempo em que as fontes são adquiridas automaticamente de acordo o assunto desejado. Tanto para a extração da notícia como para as suas respectivas fontes, fez-se o uso de *web crawlers*, um agente que realiza a coleta e o *parser* de dados na *web*. A extração estruturada das fontes, previamente desconhecidas, foi possível através da leitura das novas *tags* semânticas do HTML5 e de metadados que são utilizados para o compartilhamento de artigos em redes sociais. Ambos, quando utilizados da forma correta, se mostraram eficientes na indicação das partes do documento, sendo portanto um meio comum de definir a informação. Já a obtenção das sementes do rastreador foi realizada através de requisições ao motor de busca do Google. Por fim foi possível identificar padrões semânticos de representação dos dados nas tecnologias envolvidas no desenvolvimento *web*, possibilitando distribuí-los de formas suscetíveis ao processamento automático.

Palavras-chave: *Web Crawler*. Requisições *web*. *Web Semântica*.

ABSTRACT

With data growth on the web, the need for tools that assist in the consumption of this information becomes greater. Among the categories of data are the news sources which has large numbers of portals available and that a particular subject can be handled by different websites. With this in mind, the aim of this study is to determine forms of structured data extraction at the same time as the sources are automatically acquired according to the desired subject. As for the extraction of news as to their respective sources, web crawlers (an agent that performs the collection and web data parser) was used. Extracting structured sources, previously unknown, was made possible by reading the new HTML5 semantic tags and metadata that are used for sharing articles on social media. Both, when used properly, proved effective at indicating the parts of the document and are therefore a means of defining the common information. The Google search engine was used to obtain the sources of the tracker. Finally, it was possible to identify semantic patterns of data representation in the technologies involved in web development, making it possible to distribute them in ways that are susceptible to automatic processing.

Keywords: Web Crawler. Web requests. Semantic Web.

LISTA DE ILUSTRAÇÕES

Figura 1 - Arquitetura de um web crawler.....	12
Figura 2 - Processo de funcionamento de um motor de busca.....	23
Figura 3 - Estimativa do tamanho do índice do Google e do Bing durante o período de 2006 e 2015.....	24
Figura 4 - Arquitetura da web.....	26
Figura 5 - Estrutura de uma mensagem HTTP.....	28
Figura 6 - Mensagem de requisição HTTP.....	29
Figura 7 - Mensagem de resposta HTTP.....	30
Figura 8 - Web de documentos.....	37
Figura 9 - Web de dados.....	37
Figura 10 - Exemplo de uma tripla RDF.....	39
Figura 11 - Interface web do Scrapy.....	44
Figura 12 - Arquitetura do protótipo.....	45
Figura 13 - Arquitetura do Scrapy.....	46
Figura 14 - Página de resultados do Google.....	48
Figura 15 - Função responsável por montar a URL de pesquisa do Google.....	48
Figura 16 - Função para extração do domínio de uma URL.....	49
Figura 17 - Utilização de meta tags Open Graph.....	50
Figura 18 - Função que retorna o valor da marcação og:description.....	50
Figura 19 - Exemplo de seções extraídas.....	51
Figura 20 - Função para extração do conteúdo das notícias.....	52
Figura 21 - Listagem de tópicos e notícias do protótipo.....	53
Figura 22 - Detalhamento de uma notícia.....	53
Figura 23 - Conteúdo não relacionado com a notícia.....	54

LISTA DE TABELAS

Tabela 1 - Relação entre fontes encontradas e utilizadas.....	55
Tabela 2 - Relação do conteúdo das notícias dos tópicos.....	55

LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i>
CR	<i>Carriage Return</i>
CSS	<i>Cascading Style Sheets</i>
DNS	<i>Domain Name System</i>
DTD	<i>Document Type Definition</i>
FTP	<i>File Transfer Protocol</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
IA	<i>Inteligência Artificial</i>
IP	<i>Internet Protocol Adress</i>
IDE	<i>Integrated Development Environment</i>
LF	<i>Line Feed</i>
NIC	<i>Network Information Center</i>
ODT	<i>Open Document Format</i>
OSI	<i>Open System Interconnection</i>
P2P	<i>Peer-to-Peer</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
SVG	<i>Scalable Vector Graphics</i>
TCP	<i>Transmission Control Protocol</i>
TLD	<i>Top-level Domain</i>
TLS	<i>Transport Layer Security</i>
URI	<i>Uniform Resource Identifier</i>
URL	<i>Uniform Resource Locator</i>
URN	<i>Uniform Resource Name</i>
W3C	<i>World Wide Web Consortium</i>
XML	<i>eXtensible Markup Language</i>

SUMÁRIO

1 INTRODUÇÃO.....	5
1.1 OBJETIVO GERAL.....	6
1.2 OBJETIVOS ESPECÍFICOS.....	6
1.3 JUSTIFICATIVA.....	7
1.4 ESTRUTURA DO TRABALHO.....	8
2 WEB CRAWLER.....	10
2.1 TIPOS DE WEB CRAWLERS.....	11
2.2 SEMENTES.....	11
2.3 CRAWLER.....	12
2.4 PARSER.....	13
2.4.1 Expressões regulares.....	13
2.4.2 Beautiful Soup.....	14
2.4.3 Python parses.....	15
2.5 POLÍTICAS DE FUNCIONAMENTO.....	15
2.5.1 Política de seleção.....	15
2.5.2 Política de revisitação.....	16
2.5.3 Política de cordialidade.....	18
2.5.4 Política de paralelização.....	18
2.6 SCRAPY.....	19
2.6.1 Selectors.....	19
3 MOTORES DE BUSCA.....	21
3.1 ARQUITETURA.....	21
3.1.1 Indexação.....	22
3.1.2 Processo de consulta.....	22
3.2 COMPARATIVO ENTRE OS MOTORES DE BUSCA EXISTENTES.....	24
4 ARQUITETURA WEB.....	26
4.1 PROTOCOLO HTTP.....	27
4.1.1 Mensagens HTTP.....	28
4.1.1.1 Requisição.....	28
4.1.1.2 Resposta.....	29
4.1.2 HTTPS.....	30
4.2 DNS.....	31

5 ESTRUTURA DA INFORMAÇÃO.....	33
5.1 HTML.....	33
5.2 XML.....	34
5.4 WEB SEMÂNTICA.....	35
5.4.1 Dados conectados.....	36
5.4.1.1 URI.....	37
5.4.1.2 RDF.....	38
5.4.2 Vocabulários.....	39
6 TRABALHOS CORRELATOS.....	40
6.1 UM ESTUDO SOBRE WEB MINING SEMÂNTICA E WEB CRAWLER.....	40
6.2 EXTRAÇÃO DE INFORMAÇÕES EM FONTES DA WEB SEMÂNTICAS, ALTAMENTE ESTRUTURADAS E SEMI-ESTRUTURADAS.....	41
6.3 ALGORITMO RASTREADOR WEB ESPECIALISTA NUCLEAR.....	41
6.4 UM SISTEMA DE COLETA DE DADOS DE FONTES HETEROGÊNEAS BASEADO EM COMPUTAÇÃO DISTRIBUÍDA.....	42
7 WEB CRAWLER PARA EXTRAÇÃO ESTRUTURADA DE DADOS.....	43
7.1 METODOLOGIA.....	43
7.1.1 Estudo das ferramentas.....	43
7.1.2 Arquitetura.....	44
7.1.3 Implementação.....	45
7.1.3.1 Framework Scrapy.....	45
7.1.3.2 Web crawler de endereços.....	47
7.1.3.3 Extração estruturada de notícias.....	49
7.2 RESULTADOS OBTIDOS.....	54
8 CONCLUSÃO.....	57
REFERÊNCIAS.....	59

1 INTRODUÇÃO

Web crawler ou rastreador web, é uma ferramenta que realiza buscas em páginas da internet, coletando e classificando seu conteúdo, seja ele texto, imagem, vídeo, entre outros. Seu uso é mais proeminente em motores de busca, como o Google, realizando a indexação de páginas para que estas apareçam nas próximas pesquisas realizadas pelos usuários e mantendo as mesmas atualizadas na base de dados (AHUJA; BAL; VARNICA, 2014, tradução nossa).

O processo de coleta na web se dá por meio de *links* pré-definidos, também conhecidos como sementes. Entre os componentes de um rastreador para realizar a referida coleta, estão o *crawler* e o *parser*. Conforme explica Bastos (2006), o *crawler* é o responsável pela aquisição do conteúdo do endereço fornecido, que inicialmente se dá pelas sementes, enquanto o *parser* classifica os dados coletados conforme a sua relevância. A partir da varredura do conteúdo desses *links* iniciais, os novos endereços que são encontrados são armazenados para as próximas análises.

Segundo Ahmadi-abkenari e Selamat (2010, tradução nossa), existem dois tipos de *web crawlers*, os não focados e os focados. No primeiro, o rastreador tem por finalidade varrer toda a *web*, construindo assim uma grande base de informação para usos gerais, enquanto que na segunda estratégia, o escopo de busca se limita a um determinado assunto. Dessa forma o desempenho de um rastreador focado está diretamente relacionado com a qualidade das sementes fornecidas, no qual são voltadas ao conteúdo que o *crawler* deseja extrair (BATSAKIS; PETRAKIS; MILIOS, 2009, tradução nossa).

Com o crescente aumento de dados na internet, e conseqüentemente de novos veículos de informação, a aplicação de um *web crawler* focado para a busca de notícias, cuja as fontes de dados se apresentam de forma dispersa entre os diferentes veículos de comunicação online, requer uma forma dinâmica para a aquisição das bases de pesquisa, em virtude de que, segundo Furtado et al. (2015), os links encontrados a partir das sementes que não são sub endereços do próprio site, no geral são irrelevantes, como propagandas, entre outros. Se tratando de notícias, um determinado assunto pode estar disponível em diferentes grupos de sites, tornando a prévia informação das sementes algo ineficaz quando se busca um sistema flexível.

Aliado a dificuldade de obter as sementes de maneira automática, têm-se o desafio do agente rastreador ser capaz de capturar dentro do corpo da página em análise, somente o que é pertinente a notícia, ou seja, capturar os dados de forma estruturada, como por exemplo reconhecer o que é o título e o conteúdo da matéria. Ainda hoje, a informação na *web* não é descrita de forma satisfatória para que possa ser processada por máquinas, uma vez que, por exemplo, cada site possui uma forma diferente de categorizar o conteúdo dos elementos (*tags*, *IDs*, classes, entre outros) (SILVA; SANTOS; FERNEDA, 2013).

Essa pesquisa propõe portanto o desenvolvimento de um *web crawler* focado para extração estruturada dos dados, cuja sementes serão adquiridas de forma automática através de requisições a motores de pesquisa, sem a necessidade de informá-las previamente. Aplicando isso ao contexto de notícias, por via de regra os endereços resultantes da pesquisa já apontam direto para a página da postagem, não requerendo a busca em profundidade durante o *crawling*, melhorando seu desempenho.

1.1 OBJETIVO GERAL

Determinar formas de extração de dados em fontes de notícias heterogêneas da *web* adquiridas de forma automática.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

- a) compreender o funcionamento dos *web crawlers*;
- b) identificar relações entre os padrões encontrados na construção de páginas da *web*, com o foco em páginas de notícias, para uma extração estruturada dos dados;
- c) obter endereços resultantes de consultas em motores de busca;
- d) classificar os dados extraídos das páginas obtidas.

1.3 JUSTIFICATIVA

A quantidade de informação na internet vem crescendo a cada ano. Segundo Machado (2014), uma pesquisa feita em 2014 pela EMC, empresa líder no mercado internacional de armazenamento de dados, diz que em 2020 a quantidade de dados armazenados será seis vezes maior que no ano da pesquisa. Com isso, é visto que a aplicação de filtros de conteúdo, seja para os mais diversos interesses, se torna algo extremamente necessário. Com essa grande quantidade de dados, juntamente com a dinamicidade da internet, a aplicação de um *web crawler* que percorra toda a *web* seria algo inviável. Conforme explica Kumar e Vig (2012, tradução nossa), isso motivou a criação dos *web crawlers* focados.

Uma das vantagens de um rastreador *web* é sua flexibilidade de uso, podendo ser aplicado em praticamente qualquer fonte de dado na internet, e capturando qualquer informação, tendo seu comportamento precisamente determinado (KUMAR; VIG, 2012, tradução nossa). Não obstante, essa mesma flexibilidade traz consigo algumas desvantagens, pois um rastreador estruturado para conseguir navegar e retirar os dados desejados de um site, também se torna limitado e seu uso só será efetivo dentro deste endereço em específico.

Existem hoje várias APIs que fornecem a implementação de *web crawlers* para os mais diversos usos, com destaque para a Apifier e o DiffBot. APIs desse gênero oferecem uma gama de rastreadores, voltados para sites de compras, redes sociais, entre outros, entretanto não oferecem suporte para a obtenção de bases externas aos links originais, que devem ser informados, além do uso gratuito dessas APIs ser limitado. No contexto da extração de postagens em páginas existem também alguns *web crawlers* focados disponíveis de forma gratuita na internet, que são construídos para extração de dados de um único site, mas não sendo flexíveis ao ponto de retirar os dados estruturados de qualquer página da *web*.

Dentre as estratégias que se podem adotar em um rastreador *web*, conforme Bastos (2006), estão a Eficiência no *Crawling* e a Eficiência na Atualização, no qual estão diretamente ligadas ao desempenho do *web crawler*. A Eficiência no *Crawling* diz que as páginas mais importantes que devem ser analisadas primeiros, para no caso de sobrecarregar o armazenamento de dados, as perdas não serem significativas, enquanto a Eficiência na Atualização, refere-se ao controle da frequência de revisitas aos endereços para mantê-los atualizados e de

quais devem ser revisitados, visto que essa tarefa demanda uma grande quantidade de tempo. Através da solução proposta essas duas estratégias estariam sendo adotadas com a obtenção dinâmica das sementes, em consequência de que as páginas obtidas serão precisamente direcionadas para o conteúdo desejado, dado ao funcionamento dos motores de busca, e conseqüentemente, ter-se-ia uma quantidade reduzida de endereços, não havendo preocupação com o tempo gasto nas revisitas aos sites.

Este cenário consiste numa situação em que as bases de pesquisa se dispõem de forma separada, visto que existem uma grande quantidade de veículos de informação online. Tal variedade, embora seja também algo positivo, se apresenta muitas vezes um incômodo para o usuário, uma vez que há a necessidade de se visitar e visitar várias páginas regularmente e de forma manual em busca de novas informações a respeito do tema ou assunto de interesse no momento.

1.4 ESTRUTURA DO TRABALHO

Esta pesquisa está dividida em oito capítulos, no qual o primeiro trata-se da introdução, contendo a definição do problema com suas respectivas justificativas, bem como os objetivos do trabalho.

O segundo capítulo fala a respeito dos *web crawlers*, definindo seus tipos, formas de atuação e as práticas que se deve adotar no desenvolvimento de um rastreador. Também são mostrados ferramentas disponíveis para a construção de um *web crawler*.

No terceiro capítulo, é abordado o tema sobre motores de busca, no qual é explicado o processo de captura das páginas até a entrega dos resultados aos usuários. É apresentado também uma comparação entre os motores de busca existentes.

O quarto capítulo, se constitui da apresentação da estrutura de comunicação da web, descrevendo sua arquitetura e as formas de enviar e receber informações.

As formas de distribuição e estruturação de dados na web são descritas no quinto capítulo.

No capítulo seis são relacionados os trabalhos cujo assuntos são pertinentes ao tema desta pesquisa.

O capítulo sete, abrange o trabalho desenvolvido e os resultados do protótipo, e por fim no capítulo oito é apresentada a conclusão do trabalho.

2 WEB CRAWLER

A web hoje possui uma significável quantidade de informações distribuídas, em contraste com organizações mais tradicionais como bibliotecas, de forma descentralizada e em estruturas construídas de diferentes maneiras (AHUJA; BAL; VARNICA, 2014). Os avanços na área da tecnologia em relação a melhoria da velocidade da internet, aprimoramento nos sistemas gerenciadores de banco de dados, dispositivos de armazenamento cada vez maiores e mais baratos, entre outros, possibilitou a enorme contingência de informação existente hoje nas mais diversas áreas.

É necessário então o auxílio de ferramentas computacionais para a análise, interpretação, e gerência desses dados. Neste contexto, entra alguns conceitos como de *Data Mining*, que consiste, conforme Witten, Frank e Hall (2011, tradução nossa), em um processo de exploração em grandes bases de dados, com o objetivo de identificar padrões, fundamentá-los e projetar comportamentos futuros com base nessas informações. Falando especificamente em termos da *web*, têm-se a definição de *Web Mining*, ou mineração da *web*, para essa forma de recuperação de dados situada especificamente na internet e cuja primeira etapa de funcionamento se dá através de *web crawlers* (BALAN; PONMUTHURAMALINGAM, 2013, tradução nossa).

Um *web crawler* - também conhecido como *web spider*, *web bot*, *web indexer*, *web agent* ou *web robot* - é um agente rastreador autônomo que coleta informações de páginas da *web*, também conhecidas como documentos, baixando seu conteúdo para que este possa ser posteriormente utilizado para os mais diversos propósitos (JASANI; KUMBHARANA, 2014, tradução nossa). *Web crawlers* são flexíveis e podem ter seu comportamento precisamente determinado, sendo um importante método para a coleta de dados e o acompanhamento da rápida expansão da *web* (AHUJA; BAL; VARNICA, 2014, tradução nossa; KUMAR; VIG, 2012, tradução nossa).

Embora da existência de inúmeras formas de utilização desse agente, seu uso de maior destaque se dá em motores de busca, cuja a utilização de um rastreador é uma parte inerente à estrutura da ferramenta de pesquisa (ISWARY; NATH, 2013, tradução nossa). Neste âmbito, as atribuições do agente consistem na indexação de novas páginas, para que estas estejam disponíveis nas próximas

consultas dos usuários, e na atualização das informações das mesmas na base de dados.

2.1 TIPOS DE WEB CRAWLERS

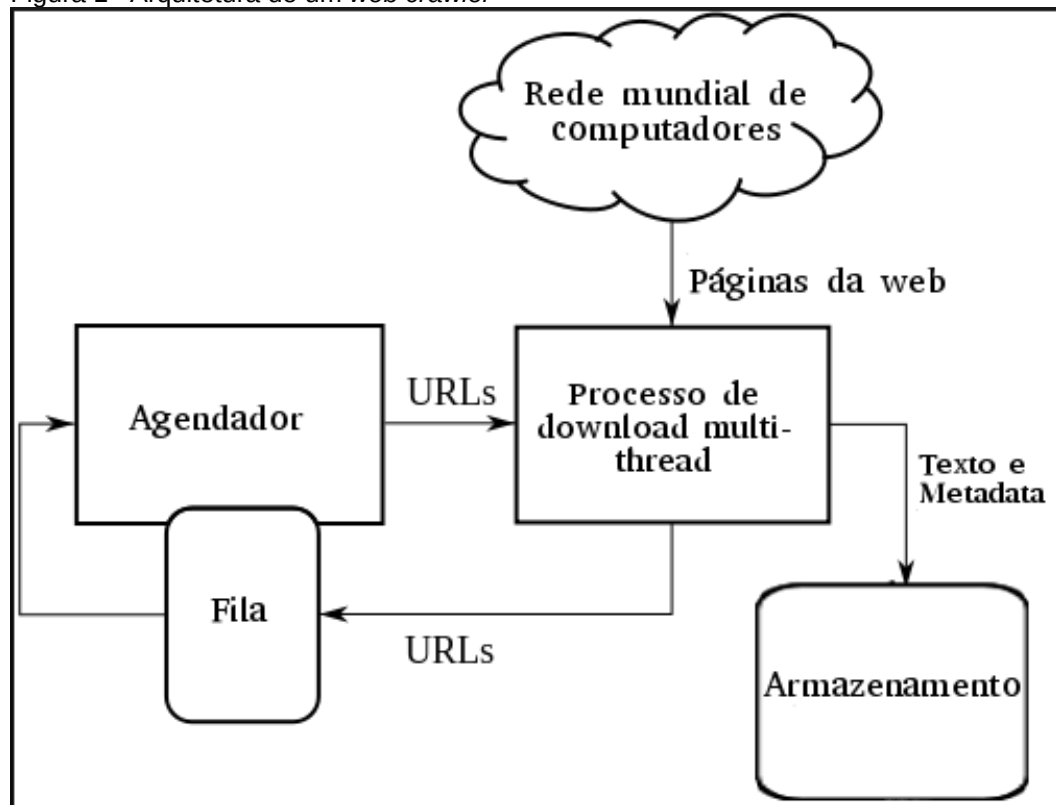
Conforme sua forma de atuação, um *web crawler* pode ser definido como não focado ou focado, ou *universal crawlers* e *topic crawlers* respectivamente. No primeiro, o rastreador tem por finalidade varrer toda a *web*, não levando em consideração seu conteúdo. Tal abordagem requer uma grande capacidade de armazenamento e gerência dos dados coletados e um maior poder computacional.

Já a segunda estratégia, os *web crawlers* focados, consiste num rastreador cujo escopo de busca se limita a uma determinada zona da *web*, filtrando somente os resultados relevantes para uma taxonomia pré-definida. Esse método elimina itens desimportantes, mantendo uma base de dados seletiva e com dimensões razoáveis (AHMADI-ABKENARI; SELAMAT, 2010, tradução nossa).

2.2 SEMENTES

O processo de *crawling* inicia com um primeiro conjunto de Localizadores Uniforme de Recursos, do inglês *Uniform Resource Locator* (URL), conhecido como sementes. Esses endereços ficam armazenados em uma lista chamada *crawl frontier*. A partir da varredura do conteúdo desses *links* iniciais, os dados das páginas em questão são armazenados e os novos endereços encontrados são colocados no *crawl frontier* e o processo é repetido até que determinada regra de parada seja atingida. No *crawl frontier* se encontra também as regras de prioridade do uso das URLs, bem como a definição de quais serão utilizadas (MCCOWN; NELSON, 2006, tradução nossa). O processo descrito se encontra na figura 1.

Se tratando de *web crawlers* focados, a escolha das sementes deve ser criteriosa, uma que vez que irá refletir diretamente no desempenho do rastreador, tanto em termos de qualidade dos resultados, quanto na questão do tempo de rastreio, que será reduzido dado o menor campo de busca.

Figura 1 - Arquitetura de um *web crawler*

Fonte: Adaptado de Castillo (2004, tradução nossa).

2.3 CRAWLER

Dentre os componentes de um rastreador está o *crawler*, que segundo Bastos (2006), é o responsável pela aquisição do conteúdo do endereço fornecido. Esta aquisição é feita através de requisições às fonte de dados. *Web crawlers* geralmente trafegam através do protocolo de transferência de hipertexto, do inglês *Hypertext Transfer Protocol* (HTTP), podendo também atuar sobre o protocolo para transferência de arquivos, do inglês *File Transfer Protocol* (FTP), protocolo de transferência de correio simples, do inglês *Simple Mail Transfer Protocol* (SMTP), utilizado no envio de e-mails, entre outros. Os dois processo básicos do *crawler* consistem na requisição e extração do conteúdo da fonte fornecida, sendo facultativas as etapas de indexação e persistência dos dados, que são relacionadas aos objetivos do rastreador, podendo ou não existir (RIBEIRO; COSTA, 2013).

Embora não haja nenhuma imposição de restrição quanto ao que pode ser ou não rastreado em um endereço, a maioria dos sites oferece um recurso para informar aos robôs quais áreas são restritas. Tal recurso é um arquivo de texto

chamado *robots.txt*. A obediência às regras descritas neste arquivo, porém, são de caráter opcional, sendo somente considerado uma boa prática respeitar tais condições (LAWSON, 2015, tradução nossa).

2.4 PARSER

Bastos (2006) também cita como segundo componente essencial de um rastreador o *parser*, que é o elemento responsável pela classificação do conteúdo baixado de uma página da *web*. Mais especificamente, o *parser* reconhece a estrutura lógica de elementos de uma página, tais como cabeçalho, corpo, *tags*, entre outros, através da análise sequencial de *tokens* do texto do documento em análise e conseqüentemente identifica falhas em sua estrutura (CROFT; METZLER; STROHMAN, 2011, tradução nossa).

Existem diferentes formas e ferramentas para se realizar a análise de documentos, sendo a linguagem de programação Python de grande destaque nesta atividade. Portanto a seguir, são descritos alguns meios para a realização do *parsing*, relacionando o uso de expressões regulares e de bibliotecas do Python.

2.4.1 Expressões regulares

Uma expressão regular, também conhecida como *regex* “é uma *String* especialmente formatada que descreve um padrão de pesquisa para correspondência de caracteres em outras *Strings*” (DEITEL; DEITEL, 2010, p. 536). Em um rastreador *web*, o uso de expressões regulares tem como principal função a separação dos elementos (*tags*, *IDs*, classes, entre outros) para serem trabalhados à parte.

Com o uso de expressões regulares é possível, por exemplo, definir um padrão genérico para o formato de links em páginas da *web* no qual faz o uso da *tag* `<a>` e do atributo *href*, para então extrair todos os URLs encontrados em uma página e assim alimentar o *crawl frontier*. Embora ofereça uma forma rápida de filtrar dados, uma expressão regular mais complexa é difícil de ser construída, se torna ilegível, e mesmo sendo mais robusta qualquer mudança na alteração do *layout* de uma página *web* pode torná-la inválida (LAWSON, 2015, tradução nossa).

2.4.2 Beautiful Soup

Esta é uma biblioteca em Python, no qual oferece vários recursos tanto para a análise quanto para a extração de dados em páginas da *web*. Qualquer entrada neste formato é convertida em objetos *Beautiful Soup*, e com base nos seus diferentes atributos e *tags* é possível realizar a extração do conteúdo desejado (NAIR, 2015, tradução nossa).

Dentre os objetos existentes na biblioteca, os principais com o qual são lidados diretamente são *BeautifulSoup*, *Tag* e *NavigableString*. O primeiro, *BeautifulSoup*, é o ponto de início do uso da biblioteca, no qual representa a própria entrada que será analisada, podendo ser no formato de Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language* (HTML), ou Linguagem de Marcação Extensível, do inglês *eXtensible Markup Language* (XML). O segundo objeto trata das *tags* do documento em questão, sendo que durante o *parsing* na página, toda *tag* passa a ser representada por este objeto, que possui uma série de métodos para a busca e navegação. Por último, o *NavigableString* representa o texto dentro de uma *tag*, contendo também alguns métodos que auxiliam na identificação dos dados.

A biblioteca permite o uso de diferentes *parsers* como o `html.parser` padrão do Python, `lxml` e `html5lib`. Ao instanciar o objeto *BeautifulSoup*, caso não se informe qual *parser* usar, a biblioteca escolherá o melhor instalado, dando prioridade primeiro ao `lxml`, depois ao `html5lib`, e por último ao *HTML parser* interno do Python. O único *parser* para arquivos XML suportado atualmente é o `lxml` (RICHARDSON, [2015], tradução nossa).

Dentre as vantagens da biblioteca estão seu recurso de *encodings* para conversão de caracteres especiais para *unicode*, uma vez que nem todas as páginas da *web* especificam corretamente sua codificação, e a já citada existência de vários métodos de busca em documentos, como por exemplo o método *find* e seus similares, que com a simples informação de uma *tag* por parâmetro retorna todas as ocorrências desta com seu respectivo conteúdo. A construção para realizar tanto o *parser* como o *crawling* dos dados é mais clara e fácil do que expressões regulares, e pequenas alterações no layout de uma página não comprometerão seu funcionamento.

2.4.3 Python parses

A linguagem de programação Python é muito utilizada na área de recuperação de dados na *web*, ao mesmo tempo que é uma linguagem de fácil leitura do código e possui uma comunidade grande e ativa. Dentre os *parsers* existentes disponíveis os principais são o `html.parser`, `lxml`, e o `html5lib` (HANRETTY, 2013, tradução nossa; RICHARDSON, 2015, tradução nossa).

Dentre eles, o que apresenta maior velocidade é o `lxml`, seguido pelo `html.parser` e por último, o `html5lib`. Este, apesar de sua lentidão, utiliza técnicas de *parser* que fazem parte do padrão do HTML5. O `lxml` suporta tanto documentos HTML como em XML. O fato de ser desenvolvido em C justifica sua maior velocidade. O `lxml` pode também fazer o uso de alguns recursos do *Beautiful Soup*, como seu suporte para detecção de codificação, através do uso do módulo `soupparser`, e também se beneficiar dos recursos do `html5lib` através do módulo `html.html5parser` (RICHARDSON, 2015, tradução nossa).

2.5 POLÍTICAS DE FUNCIONAMENTO

A *web* apresenta grandes desafios para os rastreadores, tais como sua dimensão, sua dinamicidade na geração de páginas e na velocidade de atualização. Para contornar esses problemas, um *web crawler* deve seguir um conjunto de políticas para que traga os melhores resultados, no qual serão apresentadas a seguir (ISWARY; NATH, 2013, tradução nossa).

2.5.1 Política de seleção

A Política de seleção diz respeito a utilização de métricas para classificar as páginas a serem rastreadas. Dado o tamanho da *web* e sua alta taxa de mudança, devem ser priorizadas no *crawling* as páginas de maior relevância para os objetivos do rastreador (CASTILLO; BAEZA-YATES, [200-?], tradução nossa). Para classificar tais páginas devem ser seguidos alguns parâmetros. Tendo uma página da *web* P , é possível definir a importância desta página, $I(P)$, através das seguintes métricas:

- a) *backlink count*: este parâmetro classifica a importância de uma página de acordo com seu número de citações em outros locais da *web*. Ou seja, o grau de $I(P)$ está relacionado com o número de vezes que P é referenciada em outras páginas. Para representar esta métrica se utiliza $IB(P)$ (CHO, 2001, tradução nossa);
- b) *pagerank*: este algoritmo foi desenvolvido pelos fundadores da Google, Larry Page e Sergey Brin, no qual, semelhante ao *Backlink Count*, classifica a relevância de uma página de acordo a quantidade de apontamentos em outros sites. A diferença do *PageRank* para o *Backlink Count* é que neste último, todo site referenciador possui o mesmo grau de importância, enquanto que no algoritmo da Google, é analisado também a relevância deste. Ou seja, um *link* terá um melhor grau de avaliação no *PageRank* caso seja citado pelo site da Microsoft por exemplo (BRIN; PAGE, 1998, tradução nossa);
- c) *forward link count*: o grau de importância de uma página conforme esta métrica, $IF(p)$, é calculado pela quantidade de *links* externos existentes dentro dela. Quanto maior o número de *links*, melhor classificada ela será (CHO, 2001, tradução nossa);
- d) *location metric*: por último, a *Location Metric*, representada por $IL(p)$, classifica uma página conforme o seu domínio, não levando em conta seu conteúdo. Por exemplo, conforme o autor, uma página terminada com “.com” tende a ser mais interessante do que com outras terminações menos utilizadas, ou links que contém muitas barras (“/”) também tendem a ser piores classificados dos que contém menos. Domínios contendo palavras relacionadas aos tópicos do *web crawler* também possuem um bom grau de avaliação por essa métrica (CHO, 2001, tradução nossa).

2.5.2 Política de revisitação

Conforme já citado, a *web* possui uma grande quantidade de informações, e seu conteúdo está a todo momento se alterando. Tais eventos de mudanças podem ser caracterizados como eventos de criação, atualização e de exclusão (BAEZA-YATES; CASTILLO; SAINT-JEAN, 2004, tradução nossa). Devido a isto, um

rastreador deve seguir algumas regras para manter sua base de dados atualizada com o menor custo de trabalho.

Em termos de sincronização de páginas, Cho (2001, tradução nossa) apresenta os conceitos de *Freshness* e *Age*. No primeiro, uma base de dados é considerado mais nova quando o banco de dados possui uma quantidade a mais de elementos atualizados. Dessa forma, tendo uma base de dados *A* que contém 10 elementos atualizados num total de 20 e outra base de dados *B* que contém no total 15 elementos e que estes estejam atualizados, a base *B* é considerada mais nova que a *A*. Por atualizado entende-se aqui, que o elemento é igual a sua origem do mundo real.

Grau de *Freshness* de um elemento e_i no tempo t :

$$F(e_i; t) = \begin{cases} 1 & \text{se } e_i \text{ está atualizado no tempo } t \\ 0 & \text{senão} \end{cases}$$

Grau de *Freshness* da base de dados *S*:

$$F(S; t) = \frac{1}{N} \sum_{i=1}^N F(e_i; t)$$

No segundo conceito, o *Age*, a definição de qual base está mais atualizada é definido pela idade da informação no banco de dados. Assim sendo, utilizando-se do exemplo anterior, a base *A* é considerada mais atual no caso de ter sido sincronizada há uma semana e *B* há um ano, mesmo que *A* tenha menos elementos atualizados.

Grau de *Age* de um elemento e_i no tempo t :

$$A(e_i; t) = \begin{cases} 0 & \text{se } e_i \text{ está atualizado no tempo } t \\ t - \text{tempo de modificação de } e_i & \text{caso não} \end{cases}$$

Grau de *Age* de uma base de dados *S*:

$$A(S; t) = \frac{1}{N} \sum_{i=1}^N A(e_i; t)$$

A verificação do nível de atualização de um elemento sem que haja a necessidade de obtê-lo por completo, pode ser feita através de requisições que trazem somente metadados da página em análise, que seria o caso de uma requisição HTTP do tipo HEAD, apresentada adiante no capítulo 4, no qual é possível obter a data da última modificação do documento.

2.5.3 Política de cordialidade

Diz respeito principalmente ao controle do número de acessos a determinado site para que não haja o sobrecarregamento do mesmo. A frequência de acesso a páginas feito por *web crawlers*, se não controlada, supera em muito a capacidade de um usuário comum, tendo o risco de sobrecarregar o servidor. Para controlar isso é determinado um tempo de espera mais conhecido como *delay*, entre as requisições do robô. O tempo a ser respeitado pode estar contido no arquivo *robots.txt* do site (LAWSON, 2015, tradução nossa).

Castillo e Baeza-yates ([200-?], tradução nossa) sintetiza esta e outras regras a serem seguidas em relação a política de cordialidade por um motor de busca nos três itens a seguir:

- a) um *web crawler* deve-se identificar-se, ou seja, especificar que não se trata de um usuário. Isto serve para muitos casos, dentre eles para que a contagem de visitantes da página continue consistente, e, no caso de existir um limite de largura de banda para rastreadores, que esse limite seja então aplicado a este *web crawler*;
- b) a obediência ao, anteriormente descrito, arquivo *robots.txt*, que identifica áreas restritas dos sites, limites de banda para rastreadores, entre outros;
- c) um *web crawler* deve manter o uso de banda baixa, para que não haja então a sobrecarga dos servidores.

2.5.4 Política de paralelização

Dependendo o local em que o agente rastreador vai atuar e levando em consideração seus objetivos, um processo único de *crawling* pode levar um tempo considerável para trazer os resultados esperados. Desta forma, um *web crawler*

paralelo tem por objetivo de ampliar a taxa de download de páginas sem sobrecarregar um processo único (CHO; GARCIA-MOLINA, 2002, tradução nossa).

O uso da paralelização em rastreadores traz as seguintes vantagens:

- a) escalabilidade: nem sempre as taxas de *download* podem ser, dado determinadas situações, suportadas por um único *web crawler*, devido ao grande tamanho da *web*;
- b) *network-load dispersion*: um processo de *crawling* pode ser executado em diferentes redes de internet, evitando a sobrecarga de uma única rede;
- c) *network-load reduction*: cada agente paralelo recuperaria dados somente de servidores que estão no mesmo local do *crawler*. Com os dados organizados e já filtrados na fonte, a taxa de envio para um nó central custará menos banda de rede (CHO; GARCIA-MOLINA, 2002, tradução nossa).

2.6 SCRAPY

No desenvolvimento de rastreadores *web*, algumas ferramentas estão disponíveis para facilitar o gerenciamento das requisições bem como da análise dos dados extraídos. Dentre elas, está o Scrapy, um *framework* gratuito e *open-source* escrito em Python, no qual oferece um série de funções de alto nível para a implementação de *web crawlers*. Sua estrutura permite o desenvolvimento de rastreadores escaláveis, contendo funções que auxiliam em todas as etapas do funcionamento dos mesmos, desde o *parser* até o armazenamento dos dados (LAWSON, 2015, tradução nossa).

Uma das principais vantagens do Scrapy é seu agendamento e processamento assíncrono de requisições, no qual permite que uma nova requisição possa ser executada mesmo que uma outra ainda não tenha sido concluída. Isso garante a execução de rastreios muito mais velozes. Apesar disso, o Scrapy também permite o controle sobre a polidez do rastreamento, oferecendo parâmetros que configuram o atraso de download entre solicitações, limite máximo de execuções simultâneas por domínio, entre outros (SCRAPY..., 2017).

2.6.1 Selectors

Para realizar a extração dos dados, é possível utilizar junto do Scrapy bibliotecas externas para este fim, como o *Beautiful Soup* e o *lxml*, citadas neste capítulo. Porém, o Scrapy conta com um mecanismo próprio, chamado *selectors*.

Os *selectors* possuem funções para a realizar a seleção de partes de um documento HTML ou XML através de expressões Folhas de Estilo em Cascata, do inglês Cascading Style Sheets (CSS) ou XPath, podendo também utilizar expressões regulares de forma conjunta. Sua construção foi baseada no *lxml*, o que resulta em um bom desempenho (SCRAPY..., 2017).

3 MOTORES DE BUSCA

Um motor de busca, também conhecido como *search engine* é uma ferramenta que realiza a busca de informações armazenadas na rede mundial de computadores. Seu surgimento deu-se devido ao crescimento no número de *web sites* na década de 90, o que dificultava a busca de dados de forma rápida. A primeira ferramenta nesses moldes foi criada em 1990 e se chamava Archie, no qual realizava buscas de arquivos no protocolo FTP. No ano seguinte surgiram outras como o Veronica e Jughead que trabalhavam em cima do protocolo Gopher, porém não eram motores de busca na *web* propriamente ditos (SEYMOUR; FRANTSVOG; KUMAR, 2011, tradução nossa).

O primeiro motor de busca na *web* foi criado por Oscar Nierstrasz em 1993, chamado W3Catalog. A partir daí começaram a ser usados *web robots* para gerar o índice de sites, no qual, o Jump Station foi a primeira ferramenta de busca na *Word Wide Web* que combinava a três características essenciais de um motor de busca: *crawling*, indexação e pesquisa.

O motor de busca mais popular atualmente é o Google, criado em 1998 por Larry Page e Sergey Brin. À época da criação, já havia outros motores de busca em uso, porém, projetando grandes taxas de crescimento da *web*, Page e Brin viram que as páginas classificadas como relevantes não seriam somente uma pequena porção, o que implicaria em problemas, uma vez que ao realizar uma pesquisa o usuário tende a analisar somente os primeiros resultados. Para, então, aumentar a precisão dos resultados, o Google fez o uso de algumas técnicas dentre as quais a de maior destaque é o algoritmo classificador *PageRank* explicado na seção 2.5.1 deste trabalho (BRIN; PAGE, 1998, tradução nossa).

3.1 ARQUITETURA

Se tratando da arquitetura de um motor de busca, Croft, Metzler e Strohman (2011, tradução nossa) apontam duas características que a determinam, que são a eficácia e a eficiência. A primeira está ligada a qualidade, um motor de busca deve ser capaz de recuperar as informações mais relevantes dos documentos possíveis para uma determinada *query* de pesquisa. A segunda diz respeito a

velocidade, no qual os resultados devem ser entregues ao usuário o mais rápido possível.

O autor ainda especifica dois processos que ele caracteriza por funções principais dos componentes do motor de busca. O primeiro é o processo de indexação e o segundo o processo de consulta.

3.1.1 Indexação

A indexação é uma parte fundamental de um motor de busca. A partir da realização do *crawling*, ou seja, da aquisição do conteúdo dos *web* sites e de seus *links*, o conteúdo extraído é analisado pelo *parser* do rastreador que irá retirar as informações relevantes. A forma mais comum de indexação realizada por *search engines*, também presente em outros meios, é através do uso de listas invertidas, que tem por objetivo minimizar o tempo de resposta para determinada consulta (CASTILLO, 2004, tradução nossa).

Uma lista invertida é um conjunto de termos distintos de uma determinada lista de documentos. Na primeira etapa para a criação da lista, é realizado o escaneamento dos documentos, e cada termo encontrado é armazenado em um arquivo temporário contendo o número do termo juntamente com o número de seu documento de ocorrência. Na segunda etapa, é realizada o processo de inversão do arquivo temporário, sendo este ordenado pelos termos, com o ID dos documentos de ocorrência como chave secundária. Com a existência de centenas de milhões de termos distintos na internet, a implementação e uso de listas invertidas se utiliza de recursos como tabela *Hash*, árvores de pesquisa, *clusters*, e ferramentas de compressão de dados (CASTILLO, 2004, tradução nossa; HAWKING, 2006, tradução nossa).

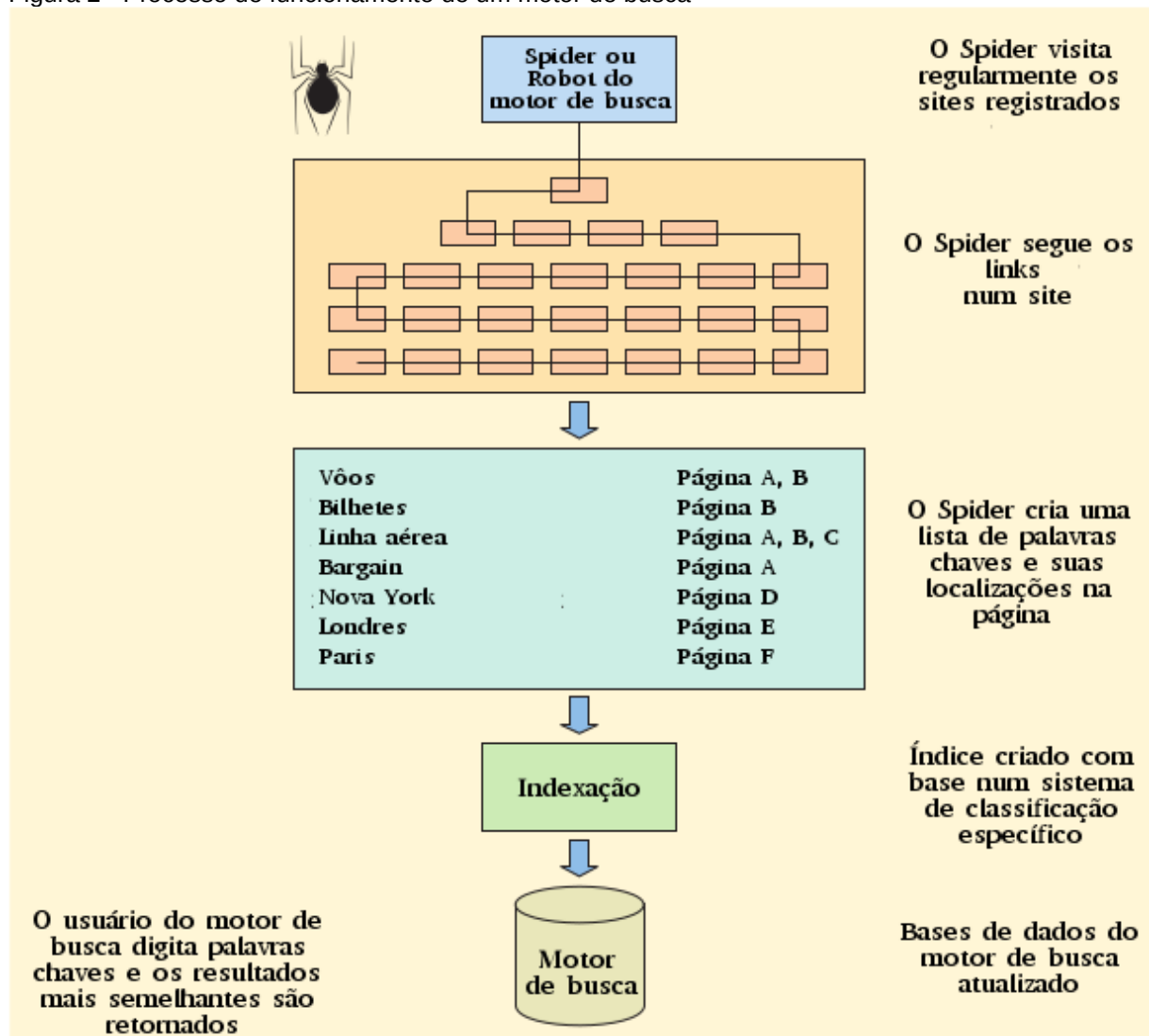
3.1.2 Processo de consulta

O processo de consulta consiste em três etapas, a Interação com o usuário, Classificação e Avaliação. A primeira fornece ao usuário uma interface para que ele realize a pesquisa, no qual, a partir da inserção da consulta no motor de busca, o seu índice é analisado e provê uma lista de páginas normalmente exibindo o título e uma parte do conteúdo. Muitos motores de busca oferecem recursos

avançados de pesquisa, tais como a busca por termos compostos e em domínios específicos (SEYMOUR; FRANTSVOG; KUMAR, 2011, tradução nossa).

Na figura 2 é sintetizado todo o processo de funcionamento de um motor de busca descrito até aqui, desde a ação do *web crawler* em capturar as páginas até a disponibilidade dos dados para os usuários. Estes, por sua vez, comumente utilizarão navegadores para acessar e fazer solicitações à ferramenta de pesquisa.

Figura 2 - Processo de funcionamento de um motor de busca

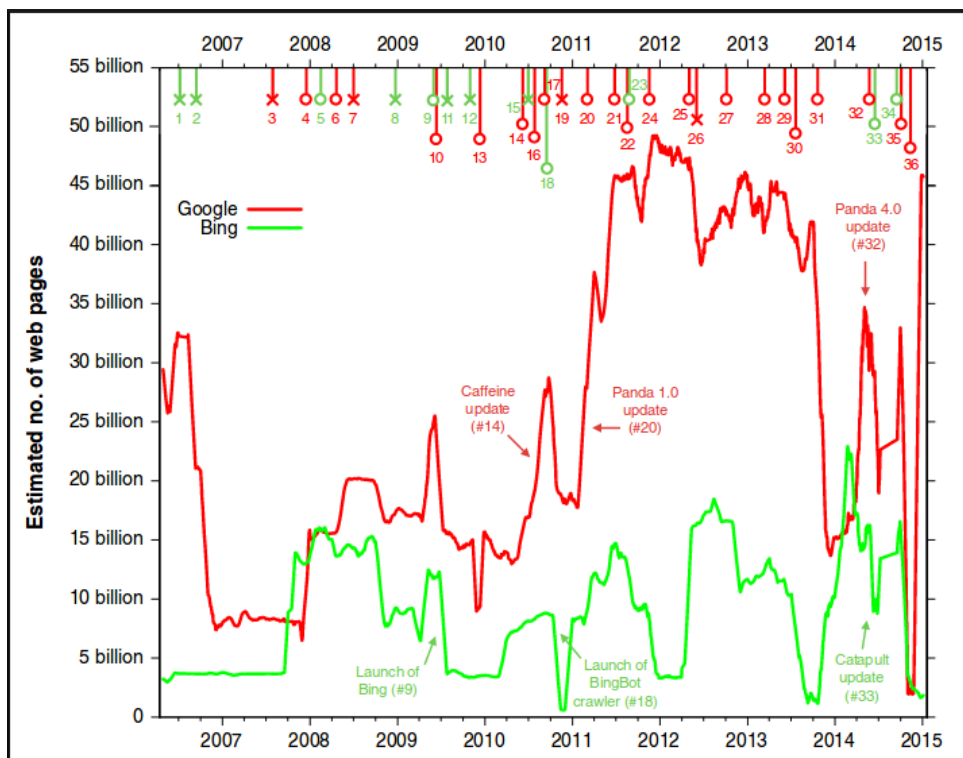


Fonte: Adaptado de Chaffey et al. (2009, tradução nossa).

3.2 COMPARATIVO ENTRE OS MOTORES DE BUSCA EXISTENTES

Um dos fatores que indica a qualidade de um motor de busca é a sua quantidade de páginas indexadas, que reflete diretamente na quantidade de informações disponíveis para os usuários. Embora seja algo difícil de estimar com precisão, Bosch, Bogers e Kunder (2016), no período de 2006 a 2015, aplicaram um método para monitorar as variações de tamanho do índice do Google e do Bing, dois dos principais motores de busca existentes. Os resultados mostraram não existir um crescimento constante no índices, ao mesmo tempo que se notou uma grande quantidade de variações, conforme mostra a figura 3. No topo do gráfico há marcações que indicam alterações na infraestrutura do motor de busca, em que se notou que as mudanças no tamanho do *index* eram influenciadas, em sua maioria, por essas alterações estruturais. Apesar disso, os maiores picos foram alcançados pelo Google chegando a até aproximadamente 50 bilhões de páginas, em comparação com o máximo de cerca de 23 bilhões do Bing. Os algoritmos ainda continuam monitorando os índices, porém novos resultados da pesquisa ainda não estão disponíveis.

Figura 3 - Estimativa do tamanho do índice do Google e do Bing durante o período de 2006 e 2015.



Fonte: Adaptado de Bosch, Bogers e Kunder (2016).

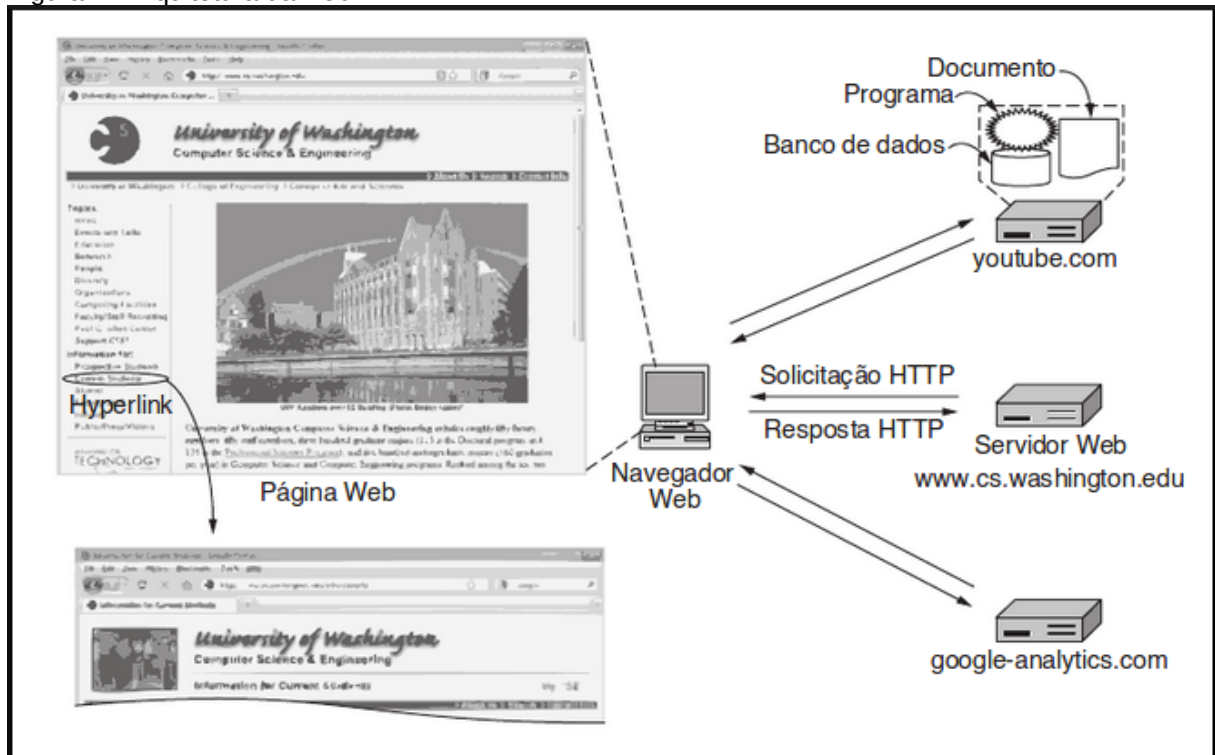
Em termos de popularidade, um ranking com dados estimados em Julho de 2017 nos Estados Unidos pelo portal eBiz, mostrou os 15 motores de busca mais populares com base nas visitas mensais. Os cinco que tiveram os resultados mais expressivos foram o Google com 1,8 bilhões de visitas mensais, seguido do Bing com 500 milhões, Yahoo! Search com 490 milhões e o Baidu com 480 milhões. Os demais não ultrapassavam os 300 milhões mensais (TOP..., 2017).

4 ARQUITETURA WEB

As duas principais arquiteturas utilizadas em aplicações em rede são a cliente-servidor e o ponto a ponto, do inglês *Peer-to-Peer* (P2P). Na arquitetura cliente-servidor há um hospedeiro dedicado, denominado servidor, sempre em funcionamento que atende requisições de outros hospedeiros, denominados clientes, enquanto que no P2P, não existe um servidor central dedicado, e a troca de informações ocorre diretamente entre cada nó da rede, denominados pares, podendo estes prover ou solicitar dados. Na primeira arquitetura citada, um exemplo comum, demonstrado na figura 4 seria um navegador web acessando determinado site, para isso deve ser realizado uma requisição ao servidor onde o site está hospedado (KUROSE; ROSS, 2013).

Ainda segundo o autor, para que exista a comunicação entre diferentes sistemas *web* é necessário que ambas as partes façam o uso do mesmo protocolo. Um protocolo definirá as regras de comunicação tais como os tipos de mensagens que deverão ser trocadas, sua sintaxe, entre outros. O principal protocolo utilizado na *web* é o HTTP.

Figura 4 - Arquitetura da web



Fonte: Tanenbaum e Wetherall (2011).

Para acessar um site, o usuário, geralmente utilizando-se de um navegador, precisa informar a este o endereço da página *web* em questão, seja na barra de endereços ou clicando em um *hiperlink* contido em outro site. As páginas na *web* são denominadas com o uso de URLs, no qual contém três informações básicas: o protocolo a ser utilizado, o endereço do servidor em que o site está hospedado, e o recurso que está sendo buscado. O segundo elemento mencionado, também conhecido como domínio, é a localização do site. Para traduzir esta informação textual no Endereço de Protocolo da Internet, do inglês *Internet Protocol Adress* (IP) da máquina servidora é utilizado o Sistema de Nomes de Domínios, do inglês, *Domain Name System* (DNS). Após ter o conhecimento do endereço, o navegador estabelece a conexão com o servidor enviando um comando para solicitar o recurso informado e exibi-lo ao usuário (TANENBAUM; WETHERALL, 2011).

4.1 PROTOCOLO HTTP

O protocolo HTTP é um protocolo de comunicação localizado na camada de aplicação do modelo Sistema Aberto de Interconexão, do inglês *Open System Interconnection* (OSI), sendo a base de toda comunicação na *web*. Ele é executado em dois programas, um cliente e outro servidor e definirá como esses clientes requisitarão os dados e como os servidores irão devolvê-los. O protocolo HTTP utiliza como protocolo de transporte o Protocolo de controle de transmissão, do inglês *Transmission Control Protocol* (TCP). A partir do momento em que o cliente inicia uma conexão TCP com o servidor, ambos acessarão o TCP por meio de sua interface *socket*, que mediará as trocas de mensagens. Dessa forma o cliente enviará mensagens de requisições para a sua interface *socket* e da mesma forma o servidor receberá as solicitações e enviará as respostas através desta interface. O HTTP não se preocupa com garantia de entrega dos dados, bem como o tratamento de possíveis erros na transmissão, sendo isto responsabilidade do TCP (KUROSE; ROSS, 2013).

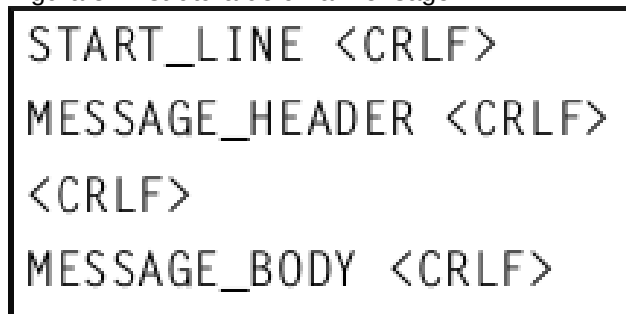
A respeito do comportamento do HTTP quanto a essas transações, é importante notar que o protocolo não retém para si as informações do cliente nos processos de requisições e respostas, tomando cada transação como independente. Ou seja, se um usuário solicitar um mesmo arquivo duas vezes em um pequeno

intervalo de tempo, não haverá nenhum aviso de que tal arquivo já foi enviado. Essa característica torna o protocolo HTTP conhecido como um protocolo sem estado ou *stateless*.

4.1.1 Mensagens HTTP

As especificações do protocolo HTTP definem dois formatos de mensagens, que são as de requisição e de resposta. A estrutura de ambos consiste basicamente, conforme mostra a figura 5, em uma linha inicial, no qual é informado o tipo de mensagem, um cabeçalho da mensagem, não obrigatório, que especifica uma coleção de parâmetros e opções, seguido de uma linha em branco. Após esta linha em branco, vem o corpo da mensagem, geralmente vazio quando se trata de uma requisição, que é o local onde o servidor colocará os dados solicitados. Todo fim de linha é seguido por uma nova linha, do inglês *Line Feed* (LF) e de um símbolo de retorno, do inglês *Carriage Return* (CR) (PETERSON; DAVIE, 2011, tradução nossa).

Figura 5 - Estrutura de uma mensagem HTTP



Fonte: Adaptado de Peterson e Davie (2011).

4.1.1.1 Requisição

A primeira linha de uma mensagem de requisição consiste em três informações principais. Primeiro, o método a ser utilizado, após, o local onde este método atuará, e por fim a versão do protocolo HTTP. Os principais métodos utilizados na *web* são *GET*, *HEAD*, *POST*, *PUT* e *DELETE* (KUROSE; ROSS, 2013).

Conforme explica Kurose e Ross (2013), o método *GET*, empregado pela maioria das mensagens de requisição, solicita ao servidor um objeto, que pode ser qualquer tipo de dado, sendo este objeto, bem como possíveis argumentos,

identificados na URL. Idêntico ao *GET*, o método *HEAD* também solicita um objeto, com a diferença que o servidor retornará somente o cabeçalho da resposta, omitindo o conteúdo. Nesses dois métodos o corpo da mensagem geralmente fica vazio, porém ele é utilizado no método *POST*, que é geralmente empregado em formulários, no qual são enviados dados ao servidor. Embora ainda exista a solicitação de uma página, o retorno dependerá dos dados enviados no corpo da mensagem. Já o método *PUT* é utilizado para o carregamento de objetos, enquanto que o *DELETE* permite ao usuário ou a aplicação deletar um conteúdo do servidor.

A figura 6 exemplifica uma mensagem de requisição, no qual na primeira linha é informado para ser utilizado o método *GET*, requisitando o objeto */somedir/page.html*. Os demais dados fazem parte das linhas de cabeçalho, que neste caso passam informações como o *host*, que é o hospedeiro onde está armazenado o objeto, o agente utilizado pelo usuário, neste caso um navegador Firefox, entre outros.

Figura 6 - Mensagem de requisição HTTP

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Fonte: Adaptado de Kurose e Ross (2013).

4.1.1.2 Resposta

Da mesma forma que as mensagens de requisição, uma mensagem de resposta HTTP iniciará com uma linha única. Esta linha começará com a versão do protocolo, seguida de um código de três dígitos que indica o estado da resposta e sua descrição correspondente. Este código indica se a solicitação foi bem sucedida ou não. Peterson e Davie (2011, tradução nossa) apresentam cinco grupos de código de resposta que são descritos a seguir:

- a) 1xx: informativa, indica que requisição foi recebida e o processo está em andamento;
- b) 2xx: sucesso, indica que a solicitação do usuário foi recebida, e processada com êxito;

- c) 3xx: redirecionamento, informa que ações extras devem ser tomadas para completar a requisição;
- d) 4xx: erro de cliente, indica um erro por parte do cliente, por exemplo um erro de sintaxe;
- e) 5xx: erro de servidor, indica falha no servidor ao tentar processar um solicitação aparentemente válida.

Na figura 7, é apresentada uma mensagem de resposta. Neste exemplo, na primeira linha, tem-se a versão do protocolo, seguido então do código de resposta, neste caso o 200, que entra no grupo das respostas que indicam sucesso, com a descrição *OK*. As seis linhas seguintes fazem parte da linha de cabeçalho, com informações tais como a data e hora de envio da resposta, entre outras. A seguir tem-se uma linha em branco, e após, o corpo da mensagem, que se tratando de uma mensagem de resposta, seria por exemplo, a página solicitada na mensagem de requisição.

Figura 7 - Mensagem de resposta HTTP

```

HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(dados dados dados dados dados ...)

```

Fonte: Adaptado de Kurose e Ross (2013).

4.1.2 HTTPS

O Protocolo de transferência de hipertexto seguro, do inglês *Hyper Text Transfer Protocol Secure* (HTTPS), consiste na utilização do protocolo HTTP com a criptografia do protocolo Segurança da Camada de Transporte, do inglês *Transport Layer Security* (TLS). Dessa forma os dados não são trafegados em texto puro, garantindo a segurança da informação (CLARK; VAN OORSCHOT, 2013, tradução nossa).

Conforme descrevem Dierks e Rescorla (2008, tradução nossa), o objetivo principal do TLS é fornecer a privacidade e integridade dos dados na comunicação entre duas aplicações. Uma de suas vantagens é a independência do

protocolo de aplicação, ou seja, o protocolo que utilizar o TLS continuará o mesmo, apenas adicionando esta camada de segurança, sendo responsável por gerenciar suas funcionalidades. O HTTPS portanto, é o mesmo protocolo HTTP, com uma camada extra de criptografia dos dados.

4.2 DNS

O DNS é um sistema fundamental da internet e do qual muitos serviços dependem dele. Embora também seja utilizado para obter vários tipos de informações sobre *hosts* e serviços, a atribuição principal do DNS está na obtenção de uma lista de endereços da *web* que identificam máquinas que utilizam o protocolo IP. Em outras palavras, ele traduz a informação textual do endereço de um site, para o endereço IP correspondente do servidor onde ele está armazenado (FUKUDA; SATO; MITAMURA, 2014, tradução nossa).

No início da internet quando havia um pequeno número de servidores e sites, o sistema de tradução de nomes para endereços consistia em uma tabela chamada *HOSTS.TXT*, mantida pelo Centro de Informação de Rede, do inglês *Network Information Center* (NIC). As atualizações nesse sistema eram realizadas através de solicitações de e-mails. Porém, com o crescimento da internet, este sistema se tornou inviável, surgindo em 1980 o DNS, que emprega um espaço de nome hierárquico ao invés de um espaço de nome plano e a tabela de ligações que implementa estes mapeamentos é dividida em diferentes instâncias e distribuídas pela internet. A aplicação deste banco de dados distribuído e hierárquico provem maior segurança, escalabilidade e performance (POPE et al., 2012, tradução nossa).

Sendo um sistema distribuído, nenhum servidor DNS possui todos os mapeamentos de todos os hospedeiros existentes, sendo estes registros espalhados pelos diferentes servidores ao redor do mundo. Existem três tipos de servidores DNS, que são os servidores raiz, de domínio de alto nível, do inglês *Top-level domain* (TLD) e os servidores DNS autoritativos. Quando um cliente DNS deseja determinar o IP de um site, como exemplo o *www.amazon.com*, a primeira ação é contactar um servidor raiz, que retornará os endereços IP dos servidores TLD para o domínio de alto nível *com*. Após, o servidor TLD contactado pelo cliente retornará o endereço de um servidor autoritativo para o *amazon.com* que por fim retornará o

endereço IP do portal *www.amazon.com* (PETERSON; DAVIE, 2011, tradução nossa).

5 ESTRUTURA DA INFORMAÇÃO

A internet desde seu surgimento evoluiu de forma considerável, migrando de uma versão estática para um sistema dinâmico, cuja propagação da informação ocorre de maneira rápida e em diferentes formatos. Os tipos de dados encontrados na *web* podem ser classificadas, conforme afirma Rusu et al. (2013, tradução nossa), como não estruturados, semiestruturados e estruturados. Os dados não estruturados são informações cujo modelo ou estrutura é desconhecido e que não poderiam ser armazenadas em tabelas relacionais. Vídeos, imagens, gráficos, e páginas da *web*, são considerados dados não estruturados. Já os semiestruturados, consistem em fontes estruturadas que não estão em conformidade com os modelos associados a banco de dados relacionais, porém que contém marcadores que organizam e impõem hierarquia aos elementos. Por fim os dados estruturados, são aqueles que seguem um esquema rígido predefinido para sua organização, cujo exemplo mais típico são os bancos de dados relacionais (SINT et al., 2009, tradução nossa).

A razão para a existência dessas diferentes formas de organização vem de necessidades particulares e específicas ao longo do tempo, muito embora, alguns conceitos como o de sistemas semânticos, que veremos adiante, tem por objetivo unificar a distribuição da informação, trazendo soluções para as deficiências das atuais formas de organizá-la (ALONSO-RORIS et al., 2014, tradução nossa).

5.1 HTML

Grande parte da *web* é constituída de páginas HTML, sendo um arquivo base que referencia outros objetos, tais como vídeos, imagens, gráficos, entre outros. O HTML é uma linguagem de marcação mantida e recomendada pelo Consórcio World Wide Web, do inglês *World Wide Web Consortium* (W3C), e sua versão atual é o HTML5. A vantagem de uma linguagem de marcação explícita está na facilidade de se criar navegadores destinados a ela (TANENBAUM; WETHERALL, 2011).

Sua estrutura se compõe de comandos, chamados *tags*, que são os comandos de formatação da linguagem. Em sua maioria, possuem marcadores de fechamento, como por exemplo a *tag* `<html>`, que aparece no início de cada

documento HTML, e a *tag* de fechamento `</html>` que se encontra no final dos documentos. Outras podem conter atributos em sua declaração, como por exemplo a *tag* ``, em que o atributo *src* define a imagem a ser exibida (HANRETTY, 2013).

Em sua evolução, o HTML vem se esforçando para tornar-se mais do que um estruturador de documentos na web para também dar significado aos dados. Isto destacou-se mais fortemente com os elementos semânticos do HTML5, que tem por objetivo descrever claramente o significado de uma informação tanto para o usuário como para o navegador. Algumas *tags* da versão 4 da linguagem já tinham este objetivo, porém o HTML5 estabeleceu novos elementos e significados semânticos aos já existentes. Um bom exemplo são as novas *tags* `<header>` e `<footer>`, que por si só são autodescritivas e a *tag* `<article>`, que representa uma parte independente e de maior relevância dentro de um documento. Essas novas marcações substituem a utilização da *tag* `<div>` por exemplo, que não apresenta nenhum aspecto semântico. Outro elemento essencial quando se trata de extração de dados de forma estruturada, é a *tag* `<aside>`, também introduzida no HTML5, que especifica elementos separados do conteúdo principal da página, tais como barras laterais, propagandas, entre outros (HTML5..., 2017, tradução nossa).

5.2 XML

O XML também é uma linguagem de marcação baseada em texto que faz o uso de *tags*, que tem por principais objetivos identificar, organizar e armazenar dados, ao invés de especificar como eles serão exibidos, que é o caso do HTML. Sua utilização é muito expressiva, estando presente, por exemplo, em formatos de processamento de texto como o Documento de Formato Aberto, do inglês *Open Document Format* (ODT), e formatos gráficos como o Gráficos Vetoriais Escaláveis, do inglês *Scalable Vector Graphics* (SVG), sendo suportado por diversas linguagens de bancos de dados e de programação (XML..., 2015, tradução nossa).

O XML permite certa liberdade para organizar a informação, podendo ser estruturado de diferentes maneiras. Suas *tags*, por exemplo, não são pré-definidas, permitindo ao criador do documento utilizar marcadores próprios. Esta liberdade porém, acarreta em problemas no intercâmbio da informação entre aplicações diferentes, fazendo-se necessário existir acordos sobre a estrutura da informação,

tais como nome de *tags* e atributos. Para isso, a W3C desenvolveu o *XML Schema*, uma alternativa a Definição de Tipo de Documento, do inglês *Document Type Definition* (DTD), que consiste em uma linguagem baseada em XML, para a validação de documentos neste formato. Um esquema irá definir quais os elementos e atributos são válidos para um arquivo em questão (HITZLER; KRÖTZSCH; RUDOLPH, 2010, tradução nossa).

5.4 WEB SEMÂNTICA

No início do século, notou-se que o crescimento da internet desenvolveu-se mais rapidamente como um fornecedor de documentos a pessoas, ao invés de dados e informações que poderiam ser processados automaticamente, ou seja, lido e compreendido por máquinas. Ainda hoje, tendo como exemplo nas pesquisas realizadas na *web* por motores de busca, os resultados nem sempre correspondem àquilo que desejamos, uma vez que determinado termo pode ter múltiplos significados. Partindo deste cenário, surgiu o conceito de Web Semântica, proposto por Berners-lee, Hendler e Lassila (2001, tradução nossa) em um artigo na revista *Scientific America* que introduziu a ideia ao público. Nas palavras dos autores, “a Web Semântica não é uma web separada, mas uma extensão do atual, em que a informação tem um significado bem definido, permitindo que os computadores e as pessoas trabalhem em cooperação.” (BERNERS-LEE; HENDLER; LASSILA, 2001, p. 1, tradução nossa).

Dessa forma, o objetivo principal da Web Semântica é desenvolver tecnologias que tornem a informação legível também para computadores. Vale lembrar que isto não implica em uma Inteligência Artificial (IA), pois embora muitas das técnicas necessárias para a construção da Web Semântica provenham da IA, seu objetivo principal não é fazer com máquinas entendam diretamente o pensamento dos usuários, e sim exigir destes a distribuição da informação de formas passíveis ao processamento automático. Assim sendo, busca-se alcançar a interoperabilidade semântica entre sistemas, ou seja, a capacidade de diferentes sistemas trabalharem com ferramentas comuns na definição da informação (BREITMAN, 2005).

5.4.1 Dados conectados

Tradicionalmente a web trabalha com o uso de *hyperlinks* entre seus documentos, sendo a base para navegação e para o processo de *crawling*, integrando todas as informações em um único espaço. No decorrer da primeira década do século XXI, algumas empresas, tais como Google e Amazon, começaram a prover acesso aos seus dados através de Interfaces de Programação de Aplicação, do inglês *Application Programming Interface* (API). Isto levou ao desenvolvimento de *mashups*, um sistema que combina dados de diferentes fontes. Porém a maioria dessas APIs não definem identificadores únicos aos seus dados, não podendo existir *hyperlinks* de conexão entre elas, tornando-se isoladas e fazendo com que seus desenvolvedores tenham que definir um conjunto específico de dados com o qual lidar (BIZER, 2009, tradução nossa).

Com isso, anexo ao surgimento da Web Semântica, surgiu o conceito de Dados Conectados, do inglês *Linked Data*, que se caracteriza pela utilização de *links* entre informações de fontes heterogêneas, com o objetivo de criar uma *web* de dados (BIZER; HEATH; BERNERS-LEE, 2009, tradução nossa). A *web* de dados, vem em contraponto a *web* de documentos, ou seja, a *web* atual. Hoje, os recursos são conectados por meio de Identificadores Uniforme de Recurso, do inglês *Uniform Resource Identifier* (URI), dessa forma, são os recursos que estão conectados, e não o seu conteúdo. Porém, para que as informações possam ser processadas por máquinas, é necessário que nestes mesmos recursos existam informações extras que caracterizam seu conteúdo e sua relação com os demais. Outras diferenças da *web* atual para a *web* de dados, residem na forma de acessar esses dados e da navegação entre páginas, hoje realizadas pelo acesso padrão por HTML, e por *hyperlinks* respectivamente, enquanto que na Web Semântica uma das principais tecnologias para conectar e descrever a informação é o *Framework* de Descrição de Recursos, do inglês *Resource Description Framework* (RDF), abordado posteriormente. (ISOTANI; BITTENCOURT, 2015).

Isotani e Bittencourt (2015), ilustraram a diferença entre o conceito de dados conectados e estado dominante da *web* hoje, a *web* de documentos. Na figura 9, tem-se um documento contendo *hyperlinks* para outros documentos.

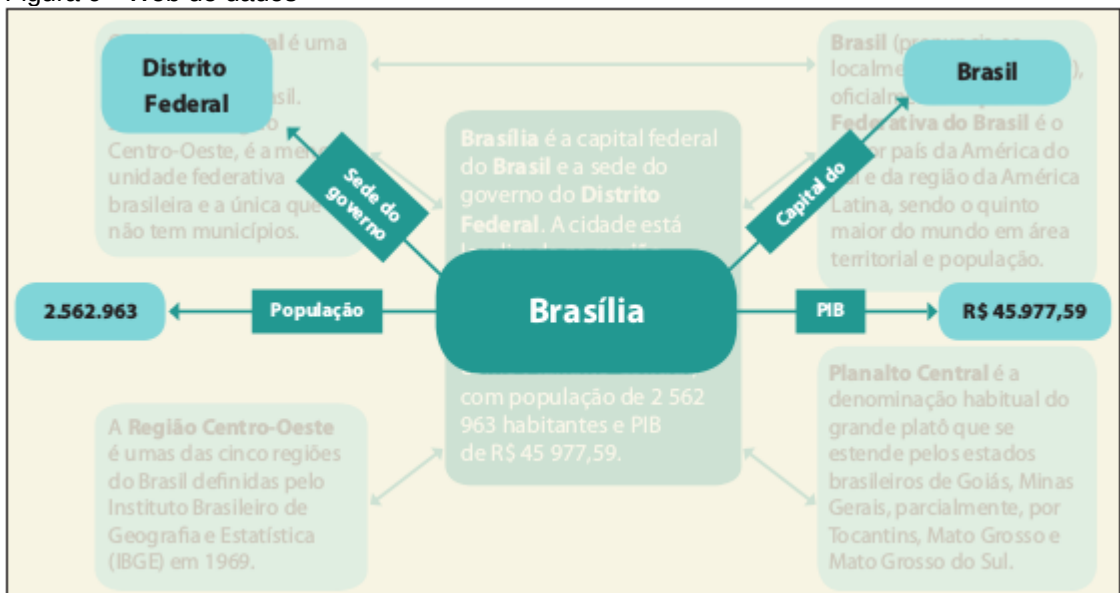
Figura 8 - Web de documentos



Fonte: Isotani e Bittencourt (2015).

Já na figura 10, são utilizados *links* RDF que contêm informações adicionais que indicam e descrevem as relações dos recursos.

Figura 9 - Web de dados



Fonte: Isotani e Bittencourt (2015)

5.4.1.1 URI

Dentre as bases da Web Semântica está a utilização de URIs, que fornece um meio simples para identificar um recurso de forma única, tanto físico

como abstrato. Um URI pode representar o nome, localização, ou ambos, fazendo com que a URL e o Nome Uniforme de Recurso, do inglês *Uniform Resource Name* (URN), sejam subconjuntos seus. O termo URL, consiste então em um subconjunto da URI que tem por objetivo identificar recursos como também fornecer meios de localizá-lo, enquanto que o URN, identifica o nome de um recurso (BERNERS-LEE; FIELDING; MASINTER, 2005, tradução nossa).

A utilização de URIs vem da necessidade de aplicar identificações uniformes a recursos, uma vez que, mesmo em tópicos relacionados, um mesmo recurso possa ser referenciado de formas diferentes, e, ao contrário disso, duas ou mais entidades diferentes podem ter o mesmo nome, sendo estas divergências eliminadas (HITZLER; KRÖTZSCH; RUDOLPH, 2010, tradução nossa).

5.4.1.2 RDF

Uma vez que os URIs buscam identificar documentos da web, como também entidades reais, o modelo RDF permite realizar inferência sobre estes recursos. Esta afirmação consiste em três informações, conhecidas como tripla, que são: sujeito, predicado e objeto. Os recursos relacionados consistem no sujeito e no objeto, e a relação existente entre esses está contida no predicado. Dessa forma o RDF fornece meios sintáticos para descrever recursos como também a descrição de suas relações (ISOTANI; BITTENCOURT, 2015).

Na tripla mencionada, o sujeito deve sempre possuir uma URI, ou seja, deve ser identificado de forma única. O predicado por sua vez, que especifica a relação por meio de propriedades, que também são recursos, deve também possuir um identificador único. Já o objeto, pode ser um recurso ou dado que se relaciona com o sujeito, dessa forma seu valor pode ser uma URI ou um literal. A sintaxe do modelo RDF é baseada em XML (CUNHA; LÓSCIO; SOUZA, 2011).

Um exemplo de uma tripla RDF é demonstrado na figura 10. Nela, tem-se três triplas fornecendo informações sobre dois recursos. Na primeira, o recurso *p91002043177* corresponde ao nome *Berna Farias*, já na segunda tripla tem-se o *ck120* como sendo a disciplina *Banco de Dados I*. Por fim a terceira tripla resume o relacionamento dos dois recursos anteriormente descritos.

Figura 10: Exemplo de uma tripla RDF

Sujeito	Predicado	Objeto
http://www.w3c.org/RDF/Validator/run/p91002043177	http://uni.org/uni/elements/1.1/nomeDocente	"Berna Farias"
http://www.w3c.org/RDF/Validator/run/ck120	http://uni.org/uni/elements/1.1/nomeDisciplina	"Banco de Dados I"
http://www.w3c.org/RDF/Validator/run/ck120	http://uni.org/uni/elements/1.1/EnsinadoPor	http://www.w3c.org/RDF/Validator/run/p91002043177

Fonte: Cunha, Lóscio e Souza (2011)

Enfim, para consultar estes dados estruturados em RDF, uma das principais tecnologias é a linguagem de consulta SPARQL. Apesar de não ser a única recomendada pela W3C para realizar tais consultas, ela apresenta algumas vantagens, uma vez que não foi criada para operar em domínio específico e é suportada por diferentes ferramentas (CUNHA; LÓSCIO; SOUZA, 2011).

5.4.2 Vocabulários

Com os dados semanticamente identificados, surge a necessidade de organizá-los em categorias, dado a proporção da *web*. Para isso, a W3C recomenda o uso de vocabulários, muitas vezes chamado de ontologias, que se caracteriza pela representação dos dados em um determinado domínio, expressando e classificando seus relacionamentos, também denominado termos (VOCABULARIES, 2015, tradução nossa).

A solução proposta através do uso de ontologias, parte da necessidade de resolver a ambiguidade de terminologias para conceitos equivalentes. Por exemplo, se uma aplicação define uma loja como sendo de *veículos*, e outra aplicação aponta para este estabelecimento como sendo uma loja de *carros*, eis aí um problema de fácil resolução para os humanos, porém não para as máquinas. Neste caso, uma ontologia, classificará tanto *carro* como *veículo* pertencentes ao mesmo domínio. Outra área beneficiada, serão os motores de busca, uma vez que as ontologias poderão prover a dissociação de palavras sinônimas (LIMA; CARVALHO, 2004).

6 TRABALHOS CORRELATOS

A área de pesquisa sobre agentes rastreadores é muito movimentada, dado que a rápida expansão da web desperta cada vez mais o interesse pela recuperação e organização das informações nela contidas. Unido a isto, almeja-se também a análise e estruturação semântica desses dados para um melhor aproveitamento.

Dessa forma, teses, dissertações e inúmeros artigos publicados em congressos, vem sendo desenvolvidos com o intuito expor o estado atual das tecnologias mencionadas e as formas de aplicação atuais e futuras.

Neste capítulo, portanto, são apresentados alguns trabalhos cuja assuntos são relacionados ao projeto proposto.

6.1 UM ESTUDO SOBRE WEB MINING SEMÂNTICA E WEB CRAWLER

O trabalho publicado por Balan e Ponmuthuramalingam (2013, tradução nossa) no *International Journal of Engineering and Computer Science*, teve como objetivo principal estudar os conceitos de mineração de dados na *web* e *web crawlers*. Dessa forma os autores se propuseram a compreender como as informações são extraídas da *web* através do uso de rastreadores e a se aprofundar nas áreas de pesquisa que dizem respeito a Web Semântica.

Com relações aos agentes rastreadores, além de explicar seu funcionamento, os autores identificaram também as técnicas utilizadas para extração, como os *web crawlers* distribuídos e os focados, apresentando suas características, objetivos, e os desafios para o desenvolvimento. Exemplificando a *Semantic Web Mining*, foi citada a relação do desenvolvimento de ambas as áreas, a Web Semântica e o *web mining*, descrevendo os motivadores que levaram ao surgimento do termo Web Semântica.

Descrevendo os meios que a Web Semântica propõe para solucionar o problema do significado da informação, foi apresentado dois fatores dificultantes neste processo, que são o tamanho da *web* e a dispersão da informação pelo fato de não haver um servidor central em que os dados ficam armazenados.

6.2 EXTRAÇÃO DE INFORMAÇÕES EM FONTES DA WEB SEMÂNTICAS, ALTAMENTE ESTRUTURADAS E SEMI-ESTRUTURADAS

Alonso-rorís et al. (2014, tradução nossa) em seu artigo, apresentaram um estudo relacionado a extração de dados em diferentes estruturas da *web*. Foram analisadas três tipos de fontes: semânticas, altamente estruturadas e semiestruturadas.

A partir da descrição desses tipos de fontes de informações pelos autores, foi analisado o formato de estruturação desses dados, bem como suas vantagens e desvantagens. Tendo esta base, foi aplicada técnicas para obter informações sobre futuros eventos educacionais em vários sites que oferecem este tipo de conteúdo. Também foi estudado a extração de recursos para utilização em aplicativos de TV digital, apresentando dois que realizam a coleta em múltiplas fontes de dados.

Em suma, concluiu-se a importância da extração automática da informação, uma vez que não há a necessidade de delegar pessoas para realizarem tal serviço, o que, perante a enorme oferta de fontes de conteúdos, acarretaria em processo lento e com resultados insatisfatórios (ALONSO-RORIS et al., 2014, tradução nossa).

6.3 ALGORITMO RASTREADOR WEB ESPECIALISTA NUCLEAR

Em sua dissertação para obtenção do grau de Mestre em Ciências na área de Tecnologia Nuclear – Reatores, Reis (2013), visou o estudo e o desenvolvimento de um *web crawler* autônomo e especialista para recuperação de conteúdo relacionado à área nuclear e seus subdomínios.

No decorrer da pesquisa o autor descreveu algumas técnicas como Redes Neurais Artificiais e Sistemas especialistas, descrevendo seus objetivos e aplicações no rastreador proposto. Vale ressaltar que o sistema foi projetado sob o modelo de um sistema especialista, tendo portanto conhecimentos prévios relacionados à área nuclear.

Para avaliação de desempenho, foi realizado experimentos de recuperação de dados com dois tópicos de pesquisa relacionados a área. Dado as métricas utilizadas para o parecer dos resultados o rastreador se mostrou preciso em cada uma delas (REIS, 2013).

6.4 UM SISTEMA DE COLETA DE DADOS DE FONTES HETEROGÊNEAS BASEADO EM COMPUTAÇÃO DISTRIBUÍDA

Em seu trabalho de conclusão de curso, para obtenção do Grau de Bacharel em Tecnologias da Informação e comunicação da Universidade Federal de Santa Catarina, Souza (2013) teve como objetivo desenvolver um rastreador para coleta de dados de forma distribuída. Através da análise das políticas de funcionamento de um *web crawler*, foi identificado a necessidade de se ter um sistema que seja flexível, que evite a sobrecarga de servidores da web e ao mesmo tempo tenha um alto desempenho em relação a capacidade de coleta.

Para atingir estes objetivos, foi proposto pelo autor foi desenvolver um *web crawler* focado no qual o trabalho de recuperação dos dados seja realizado por mais de um computador trabalhando de forma sincronizada. Por meio de um controlador, os *links* iniciais, informados manualmente, e os novos endereços encontrados a partir da análise das sementes, são distribuídos entre as máquinas pertencentes ao *cluster*. A divisão do trabalho e a comunicação entre os nós foi realizada através do uso do *framework GridGain Community Edition v4.5*.

Embora alguns problemas, como a falta de um balanceamento de carga afetaram o desempenho final do sistema, o protótipo atendeu aos objetivos. Como resultado, a partir de testes em três cenários específicos que foram, a coleta livre de dados, coleta de notícias, e coleta de documentos em PDF, foi visto que a utilização de mais de uma máquina no rastreamento trouxe resultados satisfatórios, tanto em termos de tempo como na quantidade de dados obtidos (SOUZA, 2013).

7 WEB CRAWLER PARA EXTRAÇÃO ESTRUTURADA DE DADOS

Este projeto se constituiu no desenvolvimento de uma aplicação web para agregar notícias de determinados tópicos informados pelos usuários, e monitorá-los diariamente em busca de novas informações.

Para tal, aplicando o conceito de rastreadores focados, foram desenvolvidos dois *web crawlers* que trabalham em conjunto, sendo um para a recuperação das fontes de dados, e outro para a extração dos mesmos. Uma análise nessas fontes foi necessária a fim identificar padrões e obter somente o conteúdo desejado.

7.1 METODOLOGIA

Em sua primeira parte, o projeto de pesquisa constituiu-se em um levantamento bibliográfico a fim de compreender os conceitos propostos. O início do estudo falou a respeito dos *web crawlers*, analisando seus tipos e formas de funcionamento, assim como os motores de busca, visando compreender a principal área de atuação dos rastreadores. Em seguida, foi abordado a estrutura de comunicação da web, que conseqüentemente está por trás do funcionamento dos *web crawlers*. Por último, a pesquisa definiu as formas de organização dos dados na internet, descrevendo o estado dominante e as novas propostas de distribuir e estruturar a informação na *web*, visando uma melhor interação com as máquinas.

Para a execução do projeto, primeiramente se realizou o planejamento da arquitetura, sendo definidas após, as tecnologias necessárias para atendê-la. A linguagem de programação escolhida para o desenvolvimento da parte principal do protótipo foi o Python, tanto para o *framework Scrapy* quanto para a API REST com o *Django REST Framework*, que utilizam a linguagem.

7.1.1 Estudo das ferramentas

No desenvolvimento do protótipo, para a implementação dos *web crawlers*, foi utilizado o *framework Scrapy*, versão 1.4.0. A outra parte do *back-end* da aplicação, responsável por gerenciar as requisições e delegar as chamadas aos rastreadores foi desenvolvida utilizando o *Django REST Framework*, uma biblioteca

presente no *Django* que flexibiliza a criação de *Web APIs*. Para a persistência dos dados foi utilizado o *MySQL* e para camada de aplicação se utilizou o *AngularJS*. A versão do Python utilizada foi a 2.7.

Para realizar as chamadas aos *web crawlers* através do *Django REST Framework*, foi utilizado o Scrapy, uma pequena API que fornece um servidor para que os *spiders* desenvolvidos com o Scrapy possam ser executados via requisições *web*, podendo ser chamados em paralelo. O Scrapy oferece também uma interface *web* simples, apresentada na figura 11, para monitorar os rastreadores e acessar os *logs* de execução, sendo uma ferramenta muito útil na etapa de desenvolvimento.

Figura 11 - Interface web do Scrapy

Jobs							
Go back							
Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
scrapy_news	seed_spider	6dc61d9a4aee11e8b2bd543530feac31	15746	2018-04-28 11:14:17	0:00:00		Log
Finished							
scrapy_news	seed_spider	608115984aeb11e8b2bd543530feac31		2018-04-28 10:52:23	0:00:03	2018-04-28 10:52:26	Log
scrapy_news	news_spider	62675f0c4aeb11e8b2bd543530feac31		2018-04-28 10:52:27	0:00:04	2018-04-28 10:52:31	Log
scrapy_news	seed_spider	80b73dba4aeb11e8b2bd543530feac31		2018-04-28 10:53:22	0:00:03	2018-04-28 10:53:25	Log
scrapy_news	news_spider	853bd7b04aeb11e8b2bd543530feac31		2018-04-28 10:53:27	0:00:06	2018-04-28 10:53:33	Log

Fonte: Scrapy (2017)

Como auxílio no desenvolvimento foi utilizado o PyCharm, uma interface ambiente de desenvolvimento integrado, do inglês *Integrated Development Environment* (IDE), voltada para a linguagem python.

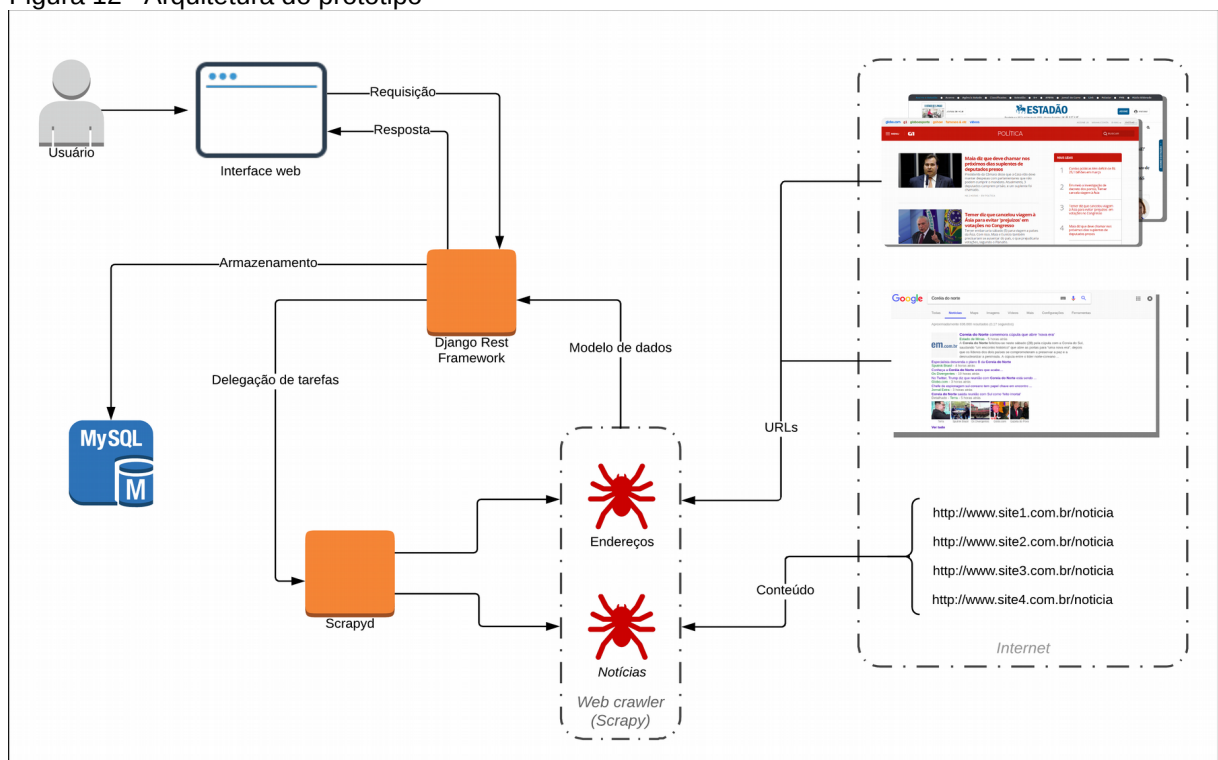
7.1.2 Arquitetura

Conforme mencionado anteriormente, para atender aos objetivos do protótipo, foram implementados dois *web crawlers*, gerenciados em conjunto. O primeiro rastreador tem como propósito obter endereços de notícias, realizando duas tarefas distintas. Na primeira, quando um novo tópico é adicionado pelo usuário, o *crawler* realiza uma requisição ao motor de pesquisa do *Google*, e na sequência faz o *parser* da resposta, capturando somente os *links* das notícias, sendo portanto as sementes. A segunda tarefa delegada a este rastreador, compreende a etapa diária de revisitação. Nesta, é obtido dos endereços adquiridos anteriormente apenas o domínio, logo a página inicial do site em questão, e em seguida é realizado uma

requisição ao mesmo, a fim de obter novas notícias relacionadas ao tópico. Os sites utilizados nas revisitas serão, portanto, sempre os que foram obtidos na primeira tarefa que é a pesquisa do Google descrita acima.

Após realizada a primeira etapa, que é a de obter os endereços do tópico informado, ou após a aquisição no processo de revisitação, é disparado uma chamada ao *web crawler* responsável por extrair o conteúdo desses endereços. Na figura 12 é mostrada a arquitetura do protótipo, com os fluxos das comunicações realizadas.

Figura 12 - Arquitetura do protótipo



Fonte: Do autor.

7.1.3 Implementação

A seguir será apresentado a etapa de implementação do protótipo, estando presentes o funcionamento do *framework* Scrapy e na sequência o detalhamento de como foi realizada a extração estruturada dos dados.

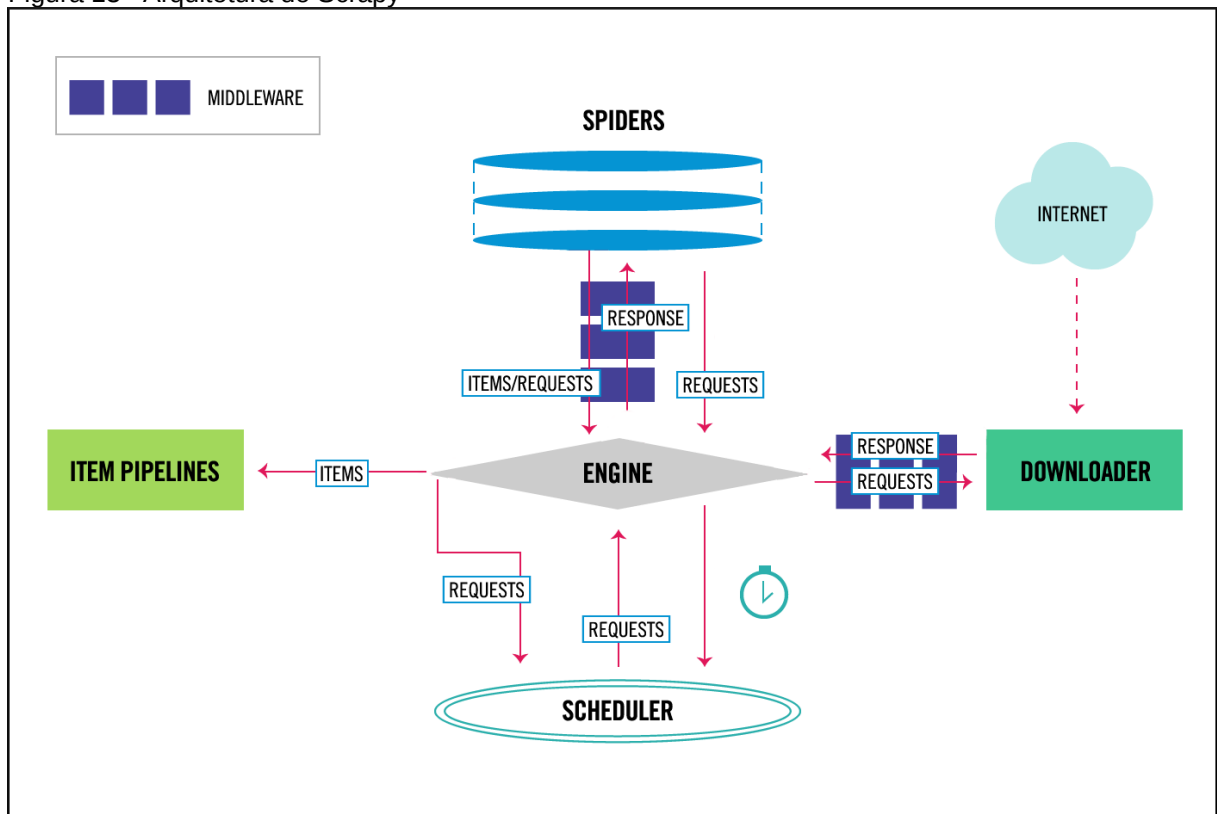
7.1.3.1 Framework Scrapy

Para realizar o *crawling* das páginas o Scrapy utiliza os objetos *Request* e *Response*, em que o primeiro representa uma requisição HTTP que é executada

pelo *Downloader*, que por sua vez retorna um objeto do tipo *Response* que contém o resultado da requisição. O gerenciamento das comunicações entre eles é realizado pelo *middleware*. Juntos, esses componentes formam o *crawler* do rastreador (SCRAPY..., 2017, tradução nossa).

Na extração das informações, o *framework* oferece uma forma de definição dos dados que é chamada de *item*. Um *item* é uma classe no qual são definidos campos que representam os dados extraídos. Após ser coletado, o *item* é enviado ao *pipeline*, uma classe que implementa uma rotina onde será realizado o seu processamento. O *pipeline* tem por objetivo avaliar os *items*, podendo realizar a persistência ou descartá-los (SCRAPY..., 2017, tradução nossa). Neste projeto portanto, o *item* é convertido em um objeto de endereço ou num objeto de notícia, conforme o *web crawler* do qual se originou. Na figura 13 é ilustrada a arquitetura de comunicação de um projeto do Scrapy como descrito previamente.

Figura 13 - Arquitetura do Scrapy



Fonte: Adaptado de Scrapy... (2017).

Na análise e extração das partes do documento, ou seja, o processo de *parser*, foi utilizado os *selectors* do Scrapy, apresentados na seção 2.6.1. O tipo de

seletor utilizado foram os seletores CSS, por serem mais legíveis e concisos. Em um único caso, que será mostrado adiante no processo de extração da notícia, fez-se necessário a utilização de expressões Xpath.

No que se refere às configurações do processo de requisição, algumas alterações foram necessárias para o correto funcionamento do projeto. A primeira diz respeito a leitura do arquivo *robots.txt*, que por padrão já é realizada pelo Scrapy de forma automática e as regras são também obedecidas. Este comportamento porém, acarretou em problemas nas requisições realizadas à página de pesquisa do Google uma vez que ela está contida no arquivo *robots.txt*, negando o acesso de agentes autônomos. Apesar de ser considerado uma boa prática respeitar essas regras, como mostrado na seção 2.6, para o prosseguimento do projeto foi necessário ignorá-las. Para isto bastou configurar o *flag* *ROBOTSTXT_OBEY* no arquivo de configurações para *False*.

Outras configurações necessárias foram a indicação do *user agent*, que é utilizado nos cabeçalhos das requisições, *pipelines*, configurações relacionadas à conexão com o projeto do Django, entre outras. Dados como tempo de atraso entre *downloads* e afins também são configurados neste arquivo, porém foram mantidos os valores padrões, sendo o *delay* entre as requisições configurado com um tempo randômico, que varia de 0.5 e 1.5 segundos.

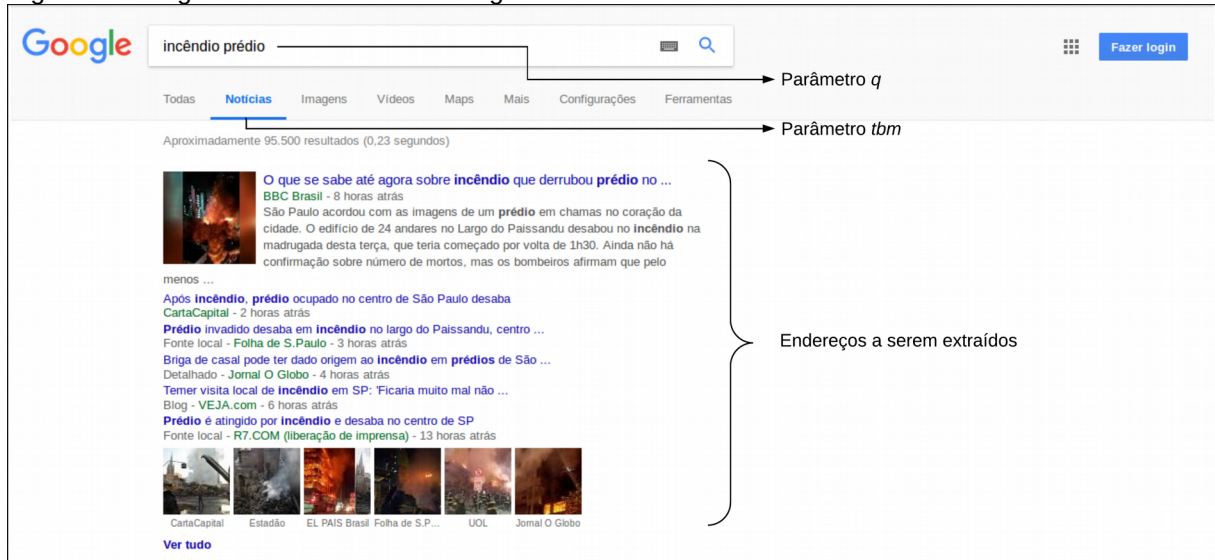
7.1.3.2 Web crawler de endereços

Na etapa de obtenção das sementes, é realizado uma requisição ao endereço de pesquisa do Google passando dois parâmetros na URL, sendo uma requisição *GET*. O primeiro parâmetro chamado *q* consiste no termo busca e o outro chamado *tbm* é referente ao filtro. Como o *Google Search* possui uma página dedicada somente a notícias, passando para o *tbm* o valor *nws*, os resultados obtidos são os daquela página.

Tomando como exemplo o termo de busca *incêndio prédio*, a URL resultante seria <https://www.google.com.br/search?q=incêndio+prédio&tbm=nws>. Na figura 14 tem-se a resposta exibida em um navegador desta requisição em questão, sendo apresentado alguns dos resultados da pesquisa, que serão as sementes do rastreador. Visando obter uma maior quantidade de resultados, a aplicação foi construída para recuperar todos os endereços encontrados nesta primeira página,

uma vez que, como será visto adiante, na extração das notícias são realizadas determinadas validações nos endereços e alguns são descartados.

Figura 14 - Página de resultados do Google



Fonte: Do autor.

Na figura 15, tem-se a função que realiza a construção do endereço de pesquisa do Google com os termos de busca informado (parâmetro *query*).

Figura 15 - Função responsável por montar a URL de pesquisa do Google

```
base_url_google = 'https://www.google.com.br/search?q={query}&tbn=nws'

def parse_google_url(self, query):
    return self.base_url_google.format(query=query.replace(" ", "+"))
```

Fonte: Do autor.

Com relação à etapa de revisitação, como citado anteriormente, é retirado dos endereços sementes apenas o domínio. Tendo como exemplo a seguinte URL <https://g1.globo.com/economia/noticia/como-dolar-mais-carro-fez-juros-na-argentina-subirem-a-40-e-qual-o-risco-para-o-brasil.ghtml> após a transformação será <https://g1.globo.com/>.

Para isso, foi utilizado a biblioteca *urlparse* do python, que divide uma URL informada em componentes permitindo gerenciá-los separadamente. A saída da função resulta na estrutura geral de uma URL: *scheme://netloc/path;parameters?query#fragment*. A biblioteca permite identificar portanto, o protocolo, nome do site, caminho do recurso, parâmetros, entre outros (URLPARSE, 2018, tradução nossa).

Na figura 16, é apresentada a função que, utilizando a *urlparse*, transforma uma lista de URLs, retornando somente o protocolo (*scheme*) e o domínio (*netloc*) de cada uma.

Figura 16 - Função para extração do domínio de uma URL

```
def parse_seeds(self, seeds):
    domain_seeds = ['{uri.scheme}://{uri.netloc}/'.format(uri=urlparse(seed.url))
                    for seed in seeds]

    return domain_seeds
```

Fonte: Do autor.

Com os domínios obtidos, deve-se agora capturar todos os *links* referentes ao assunto. Para isso, primeiramente são extraídos todos os *hyperlinks* (*tags <a>*) da página, após isso são realizados dois filtros. O primeiro seleciona somente os endereços de mesmo domínio do site em questão e que possuem o caminho de um recurso especificado, eliminando possíveis referências ao próprio site. Após isso é feita uma comparação do texto contido junto desses *hyperlinks* com o tópico de busca informado pelo usuário, e caso aquele contenha ao menos algumas das palavras deste, então é considerado um endereço pertinente à notícia.

7.1.3.3 Extração estruturada de notícias

Para o processo de extração de dados do Google com o *web crawler* de endereços detalhado anteriormente, foi implementado uma análise simples sendo necessário somente identificar a estrutura de *tags* e classes em que a página foi construída, enquanto que no *web crawler* de notícias foi necessário montar uma estrutura genérica, uma vez que as fontes de dados não são previamente conhecidas.

Na identificação dos padrões encontrados nos sites, com foco para os portais de notícias, notou-se primeiramente a utilização das marcações *Open Graph*, que são *meta tags* que otimizam o compartilhamento de conteúdo nas redes sociais, através da identificação das suas partes. Com o *Open Graph*, pode-se identificar de forma separada dados como o título, descrição, data de publicação de um artigo, além de informações do próprio site (UM GUIA..., 2018). Na figura 17 é apresentado

a definição das *meta tags* utilizando *Open Graph* da página de um artigo do site *r7.com*.

Figura 17 - Utilização de *meta tags Open Graph*

```
<meta property="og:locale" content="pt_BR">
<meta property="og:url" content="https://noticias.r7.com/saude/vacinacao-contr-a-gripe-tem-dia-d-hoje-e-mira-40">
<meta property="og:title" content="Vacinação contra a gripe tem 'Dia D' hoje e mira 40 milhões de brasileiros">
<meta property="og:site_name" content="R7.com">
<meta property="og:description" content="Desde início do ano, vírus da doença já matou 158 no País. Vacinação é">
<meta property="og:type" content="article">
<meta property="article:author" content="R7.com">
<meta property="article:section" content="Saúde">
<meta property="article:tag" content="influenza, h1n1, h3n2, gripe, campanha">
<meta property="article:published_time" content="2018-05-12T05:00:00-03:00">
<meta property="article:modified_time" content="2018-05-12T08:03:49-03:00">
<meta property="og:image" content="https://img.r7.com/images/vacina-influenza-11052018151650829">
<meta property="og:image:width" content="660">
<meta property="og:image:height" content="360">
```

Fonte: Adaptado de Lisbôa (2018).

Desta forma, para a extração do título, descrição, data de publicação e nome do site, utilizou-se os dados disponíveis nessas *meta tags*, uma vez que são disponibilizados em praticamente todos os portais de notícias. Dessas informações, somente o título possui uma alternativa quando não são encontradas as *tags Open Graph*, que seria a própria *tag <title>* do HTML, mas apesar disso, na maioria das vezes, ela traz consigo o nome do site junto do título da matéria.

Na figura 18, é apresentada uma das funções que extraem dados da *meta tag Open Graph*, sendo este exemplo a recuperação da descrição da notícia, (propriedade *og:description*) no qual se encontra um breve resumo sobre o artigo.

Figura 18 - Função que retorna o valor da marcação *og:description*

```
def get_description(dom):
    description_og = dom.css('meta[property="og:description"]::attr(content)').extract_first()
    return description_og.strip() if description_og else 'Descrição indisponível'
```

Fonte: Do autor.

Para a recuperação do conteúdo da notícia, a implementação baseou-se nas novas *tags* semânticas do HTML5 apresentadas no capítulo 5 deste trabalho. Com o estudo, chegou-se a uma estrutura geral que os portais de notícias deveriam seguir, e isso foi confirmado após, com a análise comparativa entre eles. Foi realizado portanto a leitura dos dados contidos na *tag <p>* que são filhas da marcação *<article>*, ignorando quaisquer informações pertencentes às *tags <aside>*, *<header>*, *<footer>* e *<section>* que estejam dentro do *<article>*, como demonstrado

no exemplo da figura 19, em que as partes destacadas são as que serão capturadas. Os texto contidos na imagem provém de um gerador automático.

Figura 19 - Exemplo de seções extraídas

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>...</title>
</head>
<body>
  <p>Mauris eu diam commodo, dignissim nisl quis, egestas tellus. Vivamus sit amet nibh euismod, mollis ante a, commodo risus.</p>
  <article>
    <header>Cras ac scelerisque nibh. Curabitur non fermentum lorem, vel volutpat velit. Sed consectetur leo eget tempor mattis. Sed porttitor varius ullamcorper.</header>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ornare ante ut mollis condimentum. Morbi consectetur massa ac nisi efficitur, et vehicula nunc ultrices. Maecenas vitae ex sem.</p>
    <p>Mauris vel ipsum fermentum, sodales purus ultricies, molestie est. Phasellus pulvinar, lacus in malesuada facilisis, diam enim fringilla magna, non' pulvinar magna libero vel lectus.</p>
    <div>
      <p>Cras eu ex congue, mattis elit nec, condimentum erat. Proin accumsan ullamcorper metus, nec fringilla odio blandit ac. Sed eleifend dictum lorem, eget tincidunt leo suscipit quis.</p>
    </div>
    <aside>
      <p>Nulla nec tempus velit, ut fringilla nisl. Ut orci odio, malesuada eu ullamcorper ac, ullamcorper eget justo. Nam convallis euismod orci, sit amet luctus augue ultricies eu.</p>
    </aside>
    <p>Donec ac arcu scelerisque, molestie arcu vel, interdum nisi. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In tempor, risus vel aliquam vestibulum, magna tellus mattis neque, vel ultrices enim dui nec eros. Vivamus consequat vel quam non blandit. Ut ut ex ante. Ut iaculis ut dolor ut sodales. Aliquam a velit non odio ornare ultrices ac ut tortor.</p>
  </article>
</body>
</html>

```

Fonte: Do autor.

A figura 20 apresenta a função que realiza este processo, capturando todos os `<article>` dentro da `tag <body>`, sendo o apenas o primeiro utilizado no seletor seguinte. Este por sua vez é encarregado de extrair o texto de todas as marcações `<p>` seguindo a regra de negação descrita anteriormente. Para esta expressão de negação foi necessária a utilização de um seletor XPath, devido ao fato dos seletores CSS não permitirem a negação de múltiplos elementos de uma só vez.

Figura 20 - Função para extração do conteúdo das notícias

```

def get_article_p_text(dom):
    article = dom.css('body article')
    p_news = article[0].xpath('.//p[not(ancestor::aside) '
                              'and not(ancestor::header) '
                              'and not(ancestor::footer) '
                              'and not(ancestor::section)]')

    p_contents = []

    for val in p_news:
        p_contents.append(''.join(val.css('*::text').extract()))

    return p_contents

```

Fonte: Do autor.

No *pipeline*, após a extração da notícia, é feita uma verificação se foi encontrado dados com a marcação `<p>` dentro da *tag* `<article>` e caso isso não ocorra, os dados da notícia que já foram capturados e seu respectivo endereço são descartados, não sendo a URL utilizada posteriormente no processo de revisitação. Desta forma o critério utilizado para avaliar se o site oferece ou não uma estrutura semântica na representação da informação é baseado nas informações contidas na marcação `<article>`, pois nela é que se encontra a parte mais relevante da notícia.

Na figura 21 é apresentado a parte de interação com o usuário do protótipo, sendo a listagem da esquerda referente ao tópicos escolhidos até o momento e na direita a relação de notícias do tópico selecionado. Nesta última são exibidos os dados obtidos nas meta *tags* *Open Graph*, no qual os itens em negrito são as notícias que ainda não foram visualizadas pelo usuário.

Figura 21 - Listagem de tópicos e notícias do protótipo

The screenshot displays a user interface for a news application. On the left, there is a sidebar with a search bar labeled "#Novo tópico" and a green "Adicionar" button. Below the search bar is a list of topics, each with a hashtag, a date, and a time: "#Eleições Venezuela" (22/05/2018 14:12), "#Preço gasolina" (22/05/2018 19:02), "#Eleições 2018" (22/05/2018 19:31), and "#Copa do mundo" (22/05/2018 19:57). The "#Copa do mundo" item is highlighted in blue. To the right of the sidebar is a main content area titled "Notícias". It contains three news items, each with a title, a date, and a brief description:

- Messi se apresenta à seleção argentina para a Copa do Mundo** (22/05/2018 12:45): Principal jogador da seleção da Argentina na Copa do Mundo da Rússia, Lionel Messi se apresentou à delegação nesta terça-feira (22) em Buenos Aires, após ficar na reserva na vitória por 1 a 0 do Barcelona sobre a Real Sociedad, no último domingo (20). O voo do craque chegou com atraso à
- TV anuncia contratação de Jô Soares para Copa do Mundo**: Ex-apresentador da Globo vai participar de programa do Fox Sports com especialistas de futebol durante a transmissão da Copa do Mundo da Rússia
- Moeda? Rússia lança nota de 100 rublos comemorativa da Copa do Mundo 2018**: Assim como no cartaz oficial do torneio, ex-goleiro Lev Yashin, titular da seleção do país nas Copas de 1958 a 1970, é retratado no dinheiro equivalente a cerca de R\$ 6,00
- Inglaterra anuncia Harry Kane como capitão na Copa do Mundo** (22/05/2018 09:37): Atacante e destaque do Tottenham fará sua estreia na competição. Time de Gareth Southgate está no Grupo G, com Bélgica, Tunísia e Panamá

Fonte: Do autor.

Ao clicar em um item da listagem é aberto o detalhamento da notícia, como apresentado na figura 22. Nesta tela são exibidos o título, o conteúdo que foi extraído da *tag* `<article>` e no rodapé a fonte (nome do site) com endereço para o *link* original da matéria em questão.

Figura 22 - Detalhamento de uma notícia

The screenshot shows a modal window titled "Messi se apresenta à seleção argentina para a Copa do Mundo" with a close button (X) in the top right corner. The modal contains the following text:

Principal jogador da seleção da Argentina na Copa do Mundo da Rússia, Lionel Messi se apresentou à delegação nesta terça-feira (22) em Buenos Aires, após ficar na reserva na vitória por 1 a 0 do Barcelona sobre a Real Sociedad, no último domingo (20).

O voo do craque chegou com atraso à Argentina. Por essa razão, a comissão técnica decidiu não incluir Messi no treinamento da manhã desta terça (22). Uniformizado, o camisa 10 apenas assistiu à movimentação dos demais atletas do elenco. O atacante treinará normalmente no período da tarde.

O técnico Sampaoli anunciou a lista com os 23 jogadores convocados para a Copa do Mundo na última segunda-feira (21). O comandante selecionou Paulo Dybala, mas deixou de fora Ricardo Centurión e Mauro Icardi. Outro nome ausente é o de Lautaro Martínez. Considerada a revelação do futebol argentino nesta temporada, o atacante do Racing não foi escolhido pelo técnico.

No entanto, Sampaoli também convocou outros nomes que fazem sucesso no futebol local. O goleiro Armani, do River Plate, e os meias Maximiliano Meza, do Independiente, e Cristian Pavón, do Boca Juniors, entraram na lista dos 23 escolhidos.

A Argentina estreia na Copa do Mundo no dia 16 de junho, contra Islândia, pela primeira rodada da fase de grupos. Antes disso, a equipe entra em campo no dia 29 de maio para duelo amistoso com o Haiti.

Fonte: [Folha de S.Paulo](#)

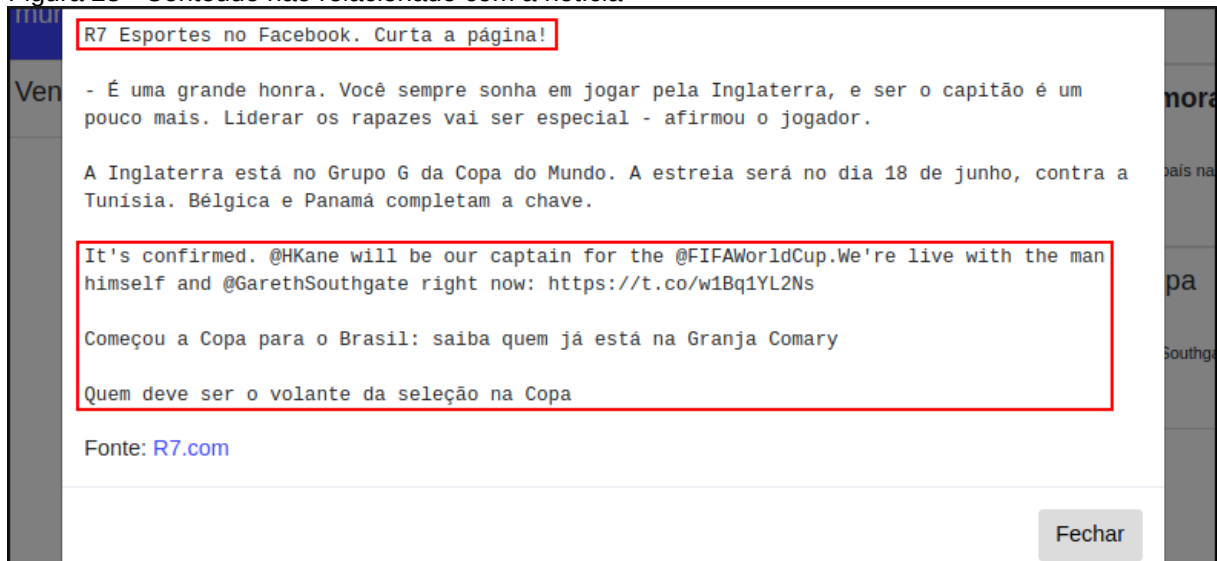
At the bottom right of the modal, there is a "Fechar" button.

Fonte: Do autor.

7.2 RESULTADOS OBTIDOS

Tratando-se da obtenção das fontes de dados, ou seja, das sementes do *web crawler*, os resultados foram os esperados, sendo todos relacionados com o tópico informado, dado a eficiência de indexação das páginas pelo motor de busca do Google. Com relação a extração da notícia, o protótipo realizou somente a extração do texto puro, sem a inserção de links e ignorando imagens, vídeos e outras estruturas como tabelas. Sendo assim, os resultados obtidos com a implementação genérica apresentada na seção 7.1.3.3, no qual não são levados em consideração sites sem dados nas *tags* <p> filhas da marcação <article>, foram em sua maioria satisfatórios, trazendo nesses casos somente o conteúdo relacionado a notícia conforme já apresentado na figura 22. Apesar disso, em algumas vezes, como mostrado na figura 23, o conteúdo obtido continha outras informações que não tinham relação direta com o artigo.

Figura 23 - Conteúdo não relacionado com a notícia



Fonte: Do autor.

Isso ocorreu por dois motivos. O primeiro é que realmente existem em alguns sites, textos disponíveis dentro da *tag* <article> sem qualquer indicação de que não fazem parte do conteúdo principal, diferente do que é encontrado na maioria dos portais, no qual informações semelhantes estavam dentro da marcação <aside>, o que seria o ideal nesses casos. O segundo motivo, de menor ocorrência, é relacionado ao próprio uso de um *web crawler* para acessar estes sites, pois muito

dos textos extraídos tinham relação com problemas de *scripts* da página, entre outros, mensagens estas que não apareceriam caso o site estivesse sendo acessado por navegador tradicional. Outra limitação tem relação com as legendas de fotos e vídeos, pois com a estratégia de extração desenvolvida a mídia correspondente não é incorporada, ficando um texto aparentemente sem sentido, porém que faz parte da notícia. Subtítulos que não estavam dentro das *tags* de parágrafo também não foram extraídos.

Na tabela 1 tem-se a relação de resultados para a pesquisa de três termos, sendo a primeira coluna a quantidade de endereços encontrados na primeira etapa do *web crawler* de endereços e a segunda referente as sementes válidas, ou seja, as que contém a *tag* `<article>` com dados.

Tabela 1 - Relação entre fontes encontradas e utilizadas

Tópico	Fontes encontradas	Fontes válidas
Eleições 2018	14	7
Greve caminhoneiros	24	11
Preço combustível	19	10

Fonte: Do autor.

Levando em conta os endereços utilizados para a extração, ou seja, os que não foram descartadas, tem-se na tabela 2 a relação de notícias obtidas para os mesmos tópicos da tabela 1 sendo indicado as porcentagem daqueles cujo conteúdo textual continha toda a notícia e as que continham dados irrelevantes para o tópico. Na indicação da notícia completa, não foi levado em conta a não incorporação de imagens, vídeos, e estrutura de dados como tabelas e listas que estavam contidos dentro da *tag* `<article>` mas que não foram capturados pelo *web crawler*, sendo apenas uma limitação do rastreador.

Tabela 2 - Relação do conteúdo das notícias dos tópicos

Tópico	Notícia completa	Informações além do conteúdo da notícia
Eleições 2018	100%	14,28%
Greve caminhoneiros	100%	36,36%
Preço combustível	100%	70%

Fonte: Do autor.

Com isso, foi verificado que dos sites que possuíam a *tag* `<article>` com marcações de parágrafo, todos continham dentro dela o conteúdo completo da notícia, só havendo a falta na indicação de conteúdos não relacionados. Dos casos que foram descartados, foi encontrado aqueles que continham esta marcação, porém ela não estava sendo utilizada para delimitar o artigo, mas sim, outras notícias listadas na mesma página, e foram desconsideradas devido ao fato de não possuírem *tags* `<p>` aninhadas.

Com relação a etapa de revisitação, realizada a cada 24 horas, embora algumas vezes trouxesse resultados pertinentes ao assuntos, o método aplicado se mostrou impreciso, pois mesmo que algumas das palavras do tópico estejam contidas no título de alguma notícia, isto não implica que ela tenha uma relação direta com o assunto desejado, e o contrário pode ocorrer também, ou seja, nenhuma palavra estar contida no título, mas mesmo assim a notícia ter relação com o tópico informado.

8 CONCLUSÃO

Com a *web* em crescente expansão e com sua dinamicidade na geração e atualização de novas informações, a utilização de ferramentas que facilitam o consumo dessa massa de dados se torna indispensável. Neste contexto, entra em cena o uso de agentes autônomos para capturar e gerir esses dados. A utilização desses agentes se mostra de grande valor não somente para o auxílio no consumo de informação, mas também para outras áreas importantes como o *Data Mining*.

A obtenção desses dados de maneira automática e estruturada implica porém, que essas informações sejam legíveis não somente para humanos, mas também para as máquinas, ou seja, distribuídas de forma semântica. Com a expansão da *web* essa preocupação foi crescendo, sendo apresentado já em 2001 o conceito da Web Semântica e seus objetivos para uma melhor distribuição dos dados na internet. Com este ideal, as tecnologias utilizadas para o desenvolvimento *web* estão evoluindo cada vez mais para que exista uma forma comum na definição das informações. Um exemplo disto é o HTML5, no qual introduziu marcações semânticas que possibilitaram no protótipo deste trabalho a extração estruturada do conteúdo da notícia. O advento das redes sociais fez também surgir soluções para um melhor compartilhamento de conteúdo nesses meios, como as meta *tags Open Graph*, utilizadas, ainda que as vezes de forma incompleta, por todos os portais de notícias que o protótipo analisou, e que se mostraram eficientes para a extração de metadados pelo *web crawler* desenvolvido.

Desta forma, um dos objetivos deste trabalho, que era encontrar padrões nos sites apesar da diversidade de estruturas, acabou por revelar uma certa semântica com a utilização dos recursos citados anteriormente, mesmo que de maneira incompleta. Portanto, com esse estudo, foi possível identificar que as tecnologias envolvidas no desenvolvimento *web* estão de fato tendo a preocupação da interoperabilidade, disponibilizando e aplicando técnicas para dar significado aos dados, embora o conceito de Dados Conectados, como por exemplo a utilização de RDF para descrição de recursos e de suas relações, não seja totalmente aplicado ainda.

Apesar disso, mesmo com o que as tecnologias oferecem no momento, foi visto que muitos sites ainda não utilizam todos os recursos disponíveis ou não utilizam da forma correta. Conforme apresentado nos resultados, existem ainda

muitos portais que não fazem o uso correto das novas *tags* HTML5 para delimitar o conteúdo principal, fazendo com o que uma possível extração dinâmica nesses casos seja muito custosa, uma vez que quando não existe uma marcação explícita, a única forma de obter o conteúdo desejado é somente através de técnicas de comparação avançada, o que por vezes resultaria em uma baixa precisão nos resultados, dado a gama de possibilidades de informações e estruturas existentes ao se trabalhar com fontes desconhecidas. Mesmo que se alcance bons resultados com este método, ainda assim a informação não estaria em conformidade com o conceito da Web Semântica, que visa avanços na forma de distribuir os dados.

A respeito da abordagem de implementação dos *web crawlers*, dentre as políticas aplicadas neste protótipo, a de maior destaque é a política de seleção, pois com os métodos utilizados não foi necessário em nenhum dos casos a busca em profundidade, como pode-se ver primeiramente no processo de obtenção das bases de pesquisa e posteriormente na etapa de revisitação. No primeiro, isso ocorreu devido aos resultados do motor de busca do Google serem precisos e já direcionados para a página da notícia. Aqui foi utilizado de forma indireta a métrica do *PageRank*, uma vez que este sistema para classificação das páginas está incorporado no mecanismo. Com relação ao segundo processo, a etapa de revisitação, naturalmente as últimas notícias tendem a ficar na página inicial, sendo mais uma vez desnecessário a busca em profundidade. Dentre as dificuldades encontradas pode-se citar a seleção de *tags* com negação de elementos pai, parser geral dos dados extraídos e tratamento do *encoding* das respostas.

Como ideia para trabalhos futuros pode-se citar:

- a) aplicar ontologias na etapa de revisita para realizar a correlação do tópico com termos equivalentes;
- b) realizar a extração de mídias como imagens e vídeos incorporadas na notícia;
- c) seguir links contidos no corpo das notícias e obter os dados destes, disponibilizando ao usuário mais detalhes do artigo;
- d) estudo de ferramentas que realizam a extração de dados em fontes que aplicam os conceito de Dados Conectados, como URI e RDF.

REFERÊNCIAS

- AGUSTEN, Dina; WINARTI. Analysis on Backlinks and PageRank of Automotive Company Websites Using Crawler Tool. **World of Computer Science And Information Technology Journal**, Depok, v. 3, n. 2, p.50-55, jan. 2013. Disponível em: <<https://pdfs.semanticscholar.org/8f47/0b0f8bfa2db8f1f2c70c3ff49b4ddf056d5c.pdf>>. Acesso em: 23 ago. 2017.
- AHMADI-ABKENARI, F.; SELAMAT, Ali. A Clickstream-based Focused Trend Parallel Web Crawler. **International Journal Of Computer Applications**. Bangalore, p. 1-8. nov. 2010. Disponível em: <<http://www.ijcaonline.org/volume9/number5/pxc3871866.pdf>>. Acesso em: 05 maio 2017.
- AHUJA, Mini Singh; BAL, Jatinder Singh; VARNICA. Web Crawler: Extracting the Web Data. **International Journal of Computer Trends and Technology**. Thennur, p. 132-137. jul. 2014. Disponível em: <<http://www.ijcttjournal.org/Volume13/number-3/IJCTT-V13P128.pdf>>. Acesso em: 11 junho 2017.
- ALONSO-RORIS, Víctor M. et al . Information Extraction in Semantic, Highly-Structured, and Semi-Structured Web Sources. **Polibits**, México , n. 49, p. 69-76, jun. 2014 . Disponível em <http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1870-90442014000100009&lng=es&nrm=iso>. Acesso em 01 nov. 2017.
- BAEZA-YATES, Ricardo; CASTILLO, Carlos; SAINT-JEAN, Felipe. Web Dynamics, Structure, and Page Quality. In: LEVENE, Mark; POULOVASSILIS, Alexandra. **Web Dynamics: Adapting to Change in Content, Size, Topology and Use**. Heidelberg: Springer, 2004. p. 93-109.
- BALAN, S.; PONMUTHURAMALINGAM, Dp. A Study on Semantic Web Mining And Web Crawler. **International Journal of Engineering and Computer Science**, Mandasaur, v. 2, n. 9, p.2659-2662, set. 2013. Disponível em: <<http://www.ijecs.in/issue/v2-i9/1%20ijecs.pdf>>. Acesso em: 12 ago. 2017
- BASTOS, Valeria Menezes. **Ambiente de descoberta de conhecimento na web para a língua portuguesa**. 2006. 124 f. Tese (Doutorado) - Curso de Ciências em Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <http://www.poc.ufrj.br/teses/doutorado/inter/2006/Teses/BASTOS_VM_06_t_D_int.pdf>. Acesso em: 07 maio 2017.
- BATSAKIS, Sotiris; PETRAKIS, Euripides G.m.; MILIOS, Evangelos. Improving the performance of focused web crawlers. **Data & Knowledge Engineering**, v.68, n.10, p. 1001-1013. 2009. Disponível em: <<http://dl.acm.org/citation.cfm?id=1598337>>. Acesso em: 14 maio 2017.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L.. **Uniform Resource Identifier (URI): Generic Syntax**. 2005. Disponível em: <<https://tools.ietf.org/html/rfc3986>>. Acesso em: 16 out. 2017.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora. The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities. **Scientific American**, [s. l.], v. 284, n. 5, p.1-3, maio 2001. Disponível em: <https://www.researchgate.net/publication/225070375_The_Semantic_Web_A_New_Form_of_Web_Content_That_is_Meaningful_to_Computers_Will_Unleash_a_Revolution_of_New_Possibilities>. Acesso em: 14 out. 2017.

BIZER, Christian; HEATH, Tom; BERNERS-LEE, Tim. Linked Data - The Story So Far. **International Journal On Semantic Web And Information Systems**, [s.l.], v. 5, n. 3, p.1-22, 2009. IGI Global. <http://dx.doi.org/10.4018/jswis.2009081901>. Disponível em: <<http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf>>. Acesso em: 22 out. 2017.

BIZER, Christian. The Emerging Web of Linked Data. **Ieee Intelligent Systems**, [s.l.], v. 24, n. 5, p.87-92, set. 2009. Institute of Electrical and Electronics Engineers (IEEE). <http://dx.doi.org/10.1109/mis.2009.102>. Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5286174&tag=1>>. Acesso em: 22 out. 2017.

BOSCH, Antal van Den; BOGERS, Toine; KUNDER, Maurice de. Estimating search engine index size variability: a 9-year longitudinal study. **Scientometrics**, [s.l.], v. 107, n. 2, p.839-856, 9 fev. 2016. Springer Nature. <http://dx.doi.org/10.1007/s11192-016-1863-z>. Disponível em: <http://www.dekunder.nl/Media/10.1007_s11192-016-1863-z.pdf>. Acesso em: 1 out. 2017.

BREITMAN, Karin Koogan. **Web Semântica: A internet do futuro**. Rio de Janeiro: Ltc, 2005.

BRIN, Sergey; PAGE, Lawrence. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: INTERNATIONAL WORLD-WIDE WEB CONFERENCE (WWW), 7., 1998, Brisbane. **Proceedings...**. Brisbane: Elsevier Science Publishers B. V., 1998. p. 107 – 117. Disponível em: <<http://ilpubs.stanford.edu:8090/361/1/1998-8.pdf>>. Acesso em: 07 set. 2017.

CASTILLO, Carlos; BAEZA-YATES, Ricardo. **Web Crawling**. [200-?]. Disponível em: <<http://grupoweb.upf.es/WRG/course/slides/crawling.pdf>>. Acesso em: 27 ago. 2017.

CASTILLO, Carlos. **Effective Web Crawling**. 2004. 179 f. Tese (Doutorado) - Curso de Ciência da Computação, Universidade do Chile, Santiago, 2004. Disponível em: <http://chato.cl/papers/crawling_thesis/effective_web_crawling.pdf>. Acesso em: 24 set. 2017.

CHAFFEY, Dave et al. **Internet Marketing: Strategy, Implementation and Practice**. 4. ed. [s.i]: Prentice Hall, 2009. 736 p.

CHO, Junghoo; GARCIA-MOLINA, Hector. Parallel crawlers. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 11., 2002, Honolulu. **Proceedings...** . Honolulu: Acm, 2002. p. 124 - 135. Disponível em: <<http://ilpubs.stanford.edu:8090/733/1/2002-9.pdf>>. Acesso em: 28 ago. 2017.

CHO, Junghoo. **Crawling the web: discovery and maintenance of large-scale web data.** 2001. 172 f. Tese (Doutorado) - Curso de Ciência da Computação, Universidade Stanford, Palo Alto, 2002. Disponível em: <<http://wonconsulting.com/~cho/papers/cho-thesis.pdf>>. Acesso em: 26 ago. 2017.

CLARK, J.; VAN OORSCHOT, P. C.. SoK: SSL and HTTPS. **2013 Ieee Symposium On Security And Privacy**, [s.l.], p.511-525, maio 2013. IEEE. <http://dx.doi.org/10.1109/sp.2013.41>. Disponível em: <<http://ieeexplore.ieee.org/document/6547130/>>. Acesso em: 08 out. 2017.

CROFT, W. Bruce; METZLER, Donald; STROHMAN, Trevor. **Search Engines: Information Retrieval in Practice.** [s.i]: Pearson Education, 2011. 552 p.

CUNHA, Danusa; LÓSCIO, Bernadette Farias; SOUZA, Damires. Linked Data: da Web de Documentos para a Web de Dados. In: SANTANA et al. **Livro Texto dos Minicursos ERCEMAPI.** Teresina: Sbc, 2011. p. 79-99. Disponível em: <https://www.researchgate.net/publication/267917518_Capitulo_4_Linked_Data_da_Web_de_Documentos_para_a_Web_de_Dados>. Acesso em: 28 out. 2017.

DEITEL, Paul; DEITEL, Harvey. **Java: Como programar.** 8. ed. São Paulo: Pearson Prentice Hall, 2010. 1144 p.

DIERKS, T.; RESCORLA, E.. **The Transport Layer Security (TLS) Protocol Version 1.2.** 2008. Disponível em: <<https://tools.ietf.org/html/rfc5246>>. Acesso em: 08 out. 2017.

FURTADO, João Carlos et al. Ferramenta para extração de dados semi-estruturados para carga de um big data. **Revista Brasileira de Computação Aplicada**, [s.l.], v. 7, n. 3, p.43-52, 30 out. 2015. UPF Editora. <http://dx.doi.org/10.5335/rbca.2015.3842>. Disponível em: <https://www.researchgate.net/publication/284765972_Ferramenta_para_extracao_d_e_dados_semi-estruturados_para_carga_de_um_big_data>. Acesso em: 07 maio 2017.

FUKUDA, Kensuke; SATO, Shinta; MITAMURA, Takeshi. Towards evaluation of DNS server selection with geodesic distance. **2014 Ieee Network Operations And Management Symposium (noms)**, [s.l.], p.1-8, maio 2014. IEEE. <http://dx.doi.org/10.1109/noms.2014.6838246>. Disponível em: <<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6838246>>. Acesso em: 12 out. 2017.

HANRETTY, Chris. **Scraping the web for arts and humanities.** [s.i]: University of East Anglia, 2013. Disponível em: <https://www1.essex.ac.uk/ldev/documents/going_digital/scraping_book.pdf>. Acesso em: 23 set. 2017.

HAWKING, David. Web Search Engines: Part 2. **Computer**, [s.l.], v. 39, n. 8, p.88-90, 14 ago. 2006. Institute of Electrical and Electronics Engineers (IEEE).
<http://dx.doi.org/10.1109/mc.2006.286>. Disponível em:
 <<http://ieeexplore.ieee.org/document/1673340/?part=1>>. Acesso em: 09 set. 2017.

HITZLER, Pascal; KRÖTZSCH, Markus; RUDOLPH, Sebastian. **Foundation of the Semantic Web technologies**. Nova Iorque: Taylor And Francis Group, Llc, 2010. 427 p.

HTML5 Semantic Elements. 2017. Disponível em:
 <https://www.w3schools.com/html/html5_semantic_elements.asp>. Acesso em: 07 nov. 2017.

ISOTANI, Seiji; BITTENCOURT, Igbert. **Dados Abertos Conectados: Em Busca da Web do Conhecimento**. [s.l.]: Novatec, 2015.

ISWARY, Raja; NATH, Keshab. Web crawler. **International Journal of Advanced Research In Computer And Communication Engineering**, Silchar, v. 2, n. 10, p.4009-4012, out. 2013. Disponível em:
 <http://www.ijarcce.com/upload/2013/october/50-o-Keshab_Nath_-web_crawler.pdf>. Acesso em: 27 ago. 2017.

JAIN, Ayush; DAVE, Meenu. The Role of Backlinks in Search Engine Ranking. **International Journal of Advanced Research In Computer Science and Software Engineering**, Jaipur, v. 3, n. 4, p.596-599, 04 abr. 2013. Disponível em:
 <https://www.ijarcsse.com/docs/papers/Volume_3/4_April2013/V3I4-0356.pdf>. Acesso em: 21 ago. 2017.

JASANI, Bhavin M.; KUMBHARANA, C. K.. Analyzing Different Web Crawling Methods. **International Journal of Computer Applications**, Bangalore, v. 107, n. 5, p.23-26, 05 dez. 2014. Disponível em:
 <<http://research.ijcaonline.org/volume107/number5/pxc3900000.pdf>>. Acesso em: 14 ago. 2017.

KOUZIS-LOUKAS, Dimitrios. **Learning Scrapy: Learn the art of efficient web scraping and crawling with Python**. Birmingham: Packt Publishing Ltd, 2016.

KUMAR, Mukesh; VIG, Renu. Learnable Focused Meta Crawling Through Web. In: INTERNATIONAL CONFERENCE ON COMMUNICATION, COMPUTING & SECURITY, 2., 2012, Rourkela. **Proceedings...**. Chandigarh: Procedia Technology, 2012. p. 606 - 611. Disponível em:
 <<http://www.sciencedirect.com/science/article/pii/S2212017312006184>>. Acesso em: 14 maio 2017.

KUROSE, Jim F.; ROSS, Keith W.. **Redes de computadores e a internet: uma abordagem top-down**. 6. ed. São Paulo: Pearson Education do Brasil, 2013. Tradução Daniel Vieira.

- LAWSON, Richard. **Web Scraping with Python**: Scrape data from any website with the power of Python. Birmingham: Packt Publishing, 2015. 151 p.
- LIMA, Júnio César de; CARVALHO, Cedric Luiz de. **Uma Visão da Web Semântica**. [s.i]: Instituto de Informática - Universidade Federal de Goiás, 2004. Disponível em: <http://www.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_001-04.pdf>. Acesso em: 29 out. 2017.
- LISBÔA, Gabriela. **Vacinação contra a gripe tem 'Dia D' hoje e mira 40 milhões de brasileiros**. 2018. Disponível em: <<https://noticias.r7.com/saude/vacinacao-contra-a-gripe-tem-dia-d-hoje-e-mira-40-milhoes-de-brasileiros-12052018>>. Acesso em: 21 maio 2018.
- MACHADO, André. Estudo da EMC prevê que volume de dados virtuais armazenados será seis vezes maior em 2020. 2014. Disponível em: <<https://oglobo.globo.com/sociedade/tecnologia/estudo-da-emc-preve-que-volume-de-dados-virtuais-armazenados-sera-seis-vezes-maior-em-2020-12147682>>. Acesso em: 13 maio 2017.
- MCCOWN, Frank; NELSON, Michael L.. Evaluation of Crawling Policies for a Web-Repository Crawler. In: ACM CONFERENCE ON HYPERTEXT AND HYPERMEDIA, 17., 2006, Odense. **Proceedings...** . Odense: Acm, 2006. p. 157 - 168. Disponível em: <<http://www.harding.edu/fmccown/pubs/crawling-policies-ht06.pdf>>. Acesso em: 29 abr. 2017.
- NAIR, Vineeth G.. **Getting Started with Beautiful Soup**: Build your own Web Scraper and learn all about Web Scraping with Beautiful Soup. Birmingham: Packt Publishing, 2015. 151 p.
- PETERSON, Larry; DAVIE, Bruce. **Computer Networks**: A Systems Approach. 5. ed. [s.i]: Morgan Kaufmann, 2011. 920 p.
- POPE, Michael Brian et al. The Domain Name System—Past, Present, and Future. **Communications of The Association For Information Systems**. [s. L.], p. 329-346. 01 maio 2012. Disponível em: <<http://aisel.aisnet.org/cais/vol30/iss1/21/>>. Acesso em: 12 out. 2017.
- REIS, Thiago. **Algoritmo rastreador web especialista nuclear**. 2013. 51 f. Dissertação (Mestrado) - Curso de Tecnologia Nuclear, Universidade de São Paulo, São Paulo, 2013. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/85/85133/tde-07012014-134548/pt-br.php>>. Acesso em: 11 nov. 2017.
- RIBEIRO, Rodrigo; COSTA, Luderson. Web crawler em Java: Coleta e filtragem de conteúdo da Web. **MundojÁgil**, [s. L.], v. 1, n. 59, p.42-49, jun. 2013. Disponível em: <http://www.univale.com.br/unisite/mundo-j/artigos/59_Webcrawler.pdf>. Acesso em: 19 ago. 2017.

RICHARDSON, Leonard. **Beautiful Soup Documentation**. [2015]. Disponível em: <<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>>. Acesso em: 02 set. 2017.

RUSU, O. et al. Converting unstructured and semi-structured data into knowledge. **2013 11th Roedunet International Conference**, [s.l.], p.1-4, jan. 2013. IEEE. <http://dx.doi.org/10.1109/roedunet.2013.6511736>. Disponível em: <https://www.researchgate.net/publication/261450947_Converting_unstructured_and_semi-structured_data_into_knowledge>. Acesso em: 04 nov. 2017.

SCRAPY Documentation: Release 1.5.0. Release 1.5.0. 2017. Disponível em: <<https://media.readthedocs.org/pdf/scrapy/latest/scrapy.pdf>>. Acesso em: 20 jan. 2018.

SCRAPYD. Version 1.2.0. [S.l.]: Scrapy Developers, 2017. Disponível em: <<https://scrapyd.readthedocs.io/en/1.2/index.html>>. Acesso em: 13 maio 2018.

SEYMOUR, Tom; FRANTSVOG, Dean; KUMAR, Satheesh. History of Search Engines. **International Journal of Management & Information Systems (ijmis)**, [s.l.], v. 15, n. 4, p.47-58, 12 set. 2011. Disponível em: <https://www.researchgate.net/publication/265104813_History_Of_Search_Engines>. Acesso em: 03 set. 2017.

SHREE, Vidya; R, Pooja M. A Review on Data Extraction using Web Mining Techniques. **International Journal Of Advanced Research In Computer Science And Software Engineering**, Mysuru, v. 5, n. 4, p.194-197, abr. 2016. Disponível em: <https://www.ijarcce.com/upload/2016/april-16/IJARCCE_49.pdf>. Acesso em: 04 nov. 2017.

SILVA, Renata Eleuterio da; SANTOS, Plácida Leopoldina V. A. da Costa; FERNEDA, Edberto. Modelos de recuperação de informação e web semântica: A questão da relevância. **Informação & Informação**, [s.l.], v. 18, n. 3, p.27-44, 9 out. 2013. Universidade Estadual de Londrina. <http://dx.doi.org/10.5433/1981-8920.2013v18n3p27>. Disponível em: <http://www.uel.br/revistas/uel/index.php/informacao/article/viewFile/12822/pdf_3>. Acesso em: 10 nov. 2017.

SINGH, Mahesh; NAVITA. An extended model for effective migrating parallel web crawling with domain specific crawling. **International Journal of Enhanced Research In Science Technology & Engineering (ijerste)**, Rohtak, v. 3, n. 6, p.17-20, jun. 2014. Disponível em: <http://www.erpublications.com/uploaded_files/download/download_21_06_2014_19_18_48.pdf>. Acesso em: 11 ago. 2017.

SINT, Rolf et al. Combining Unstructured, Fully Structured and Semi-Structured Information in Semantic Wikis. In: WORKSHOP ON SEMANTIC WIKIS, 4., 2009, Heraklion. **Proceedings...**. Heraklion: Editora, 2009. p. 1 - 15. Disponível em: <<http://ceur-ws.org/Vol-464/paper-14.pdf>>. Acesso em: 04 nov. 2017.

SOUZA, Pedro Henrique. **Um sistema de coleta de dados de fontes heterogêneas baseado em computação distribuída**. 2013. 67 f. TCC (Graduação) - Curso de Tecnologias da Informação e Comunicação, Universidade Federal de Santa Catarina, Araranguá, 2013. Disponível em: <<https://repositorio.ufsc.br/handle/123456789/99629>>. Acesso em: 22 abril 2017.

TANENBAUM, Andrew S.; WETHERALL, David J.. **Redes de computadores**. 5. ed. [s.i]: Campus, 2011.
TOP 15 Most Popular Search Engines | July 2017. 2017. Disponível em: <<http://www.ebizmba.com/articles/search-engines>>. Acesso em: 01 out. 2017.

UMGUIA de compartilhamento para Webmasters. 2018. Disponível em: <https://developers.facebook.com/docs/sharing/webmasters?locale=pt_BR#markup>. Acesso em: 12 maio 2018.

URLPARSE: Parse URLs into components. Parse URLs into components. 2018. Disponível em: <<https://docs.python.org/2/library/urlparse.html#module-urllparse>>. Acesso em: 05 maio 2018.

VOCABULARIES. 2015. Disponível em: <<https://www.w3.org/standards/semanticweb/ontology>>. Acesso em: 22 out. 2017.

WITTEN, Ian H.; FRANK, Eibe; HALL, Mark A.. **Data mining: Practical machine learning tools and techniques**. 3. ed. Burlington: Elsevier, 2011. 629 p.
XML Essentials. 2015. Disponível em: <<https://www.w3.org/standards/xml/core>>. Acesso em: 08 nov. 2017.

APÊNDICES

APÊNDICE A – Artigo

EXTRAÇÃO ESTRUTURADA DE DADOS EM FONTES HETEROGÊNEAS COM WEB CRAWLERS

Gustavo Fabro¹

¹Universidade do Extremo Sul Catarinense (UNESC)
Caixa Postal 3167 – 88806-000 – Criciúma – SC – Brazil

gustavofabro.f@gmail.com

Abstract. *With data growth on the web, the need for tools that assist in the consumption of this information becomes greater. Among the categories of data are the news sources which has large numbers of portals available and that a particular subject can be handled by different websites. As for the extraction of news as to their respective sources, web crawlers (an agent that performs the collection and web data parser) was used. Extracting structured sources, previously unknown, was made possible by reading the new HTML5 semantic tags and metadata that are used for sharing articles on social media. Both, when used properly, proved effective at indicating the parts of the document and are therefore a means of defining the common information. The Google search engine was used to obtain the sources of the tracker. Finally, it was possible to identify semantic patterns of data representation in the technologies involved in web development, making it possible to distribute them in ways that are susceptible to automatic processing.*

Resumo. *Com crescimento de dados na web torna-se cada vez maior a necessidade de ferramentas que auxiliam no consumo dessas informações. Dentre as categorias desses dados estão as fontes de notícias, em que há um grande número de portais disponíveis e no qual um determinado assunto pode ser tratado por diferentes sites. Tanto para a extração da notícia como para as suas respectivas fontes, fez-se o uso de web crawlers, um agente que realiza a coleta e o parser de dados na web. No protótipo desenvolvido, a extração estruturada das fontes, previamente desconhecidas, foi possível através da leitura das novas tags semânticas do HTML5 e de metadados que são utilizados para o compartilhamento de artigos em redes sociais. Já a obtenção das sementes do rastreador foi realizada através de requisições ao motor de busca do Google. Por fim foi possível identificar padrões semânticos de representação dos dados nas tecnologias envolvidas no desenvolvimento web, possibilitando distribuí-los de formas suscetíveis ao processamento automático.*

1. Introdução

Web crawler ou rastreador web, é uma ferramenta que realiza buscas em páginas da internet, coletando e classificando seu conteúdo, seja ele texto, imagem, vídeo, entre outros. Seu uso é mais proeminente em motores de busca, como o Google, realizando a indexação de páginas para que estas apareçam nas próximas pesquisas realizadas pelos usuários e mantendo as mesmas atualizadas na base de dados (AHUJA; BAL; VARNICA, 2014, tradução nossa).

O processo de coleta na *web* se dá por meio de *links* pré-definidos, também conhecidos como sementes. Entre os componentes de um rastreador para realizar a referida coleta, estão o *crawler* e o *parser*. Conforme explica Ahmadi-abkenari e Selamat (2010, tradução nossa), existem dois tipos de *web crawlers*, os não focados e os focados. No primeiro, o rastreador tem por finalidade varrer toda a *web*, construindo assim uma grande base de informação para usos gerais, enquanto que na segunda estratégia, o escopo de busca se limita a um determinado assunto.

Com o crescente aumento de dados na internet, e conseqüentemente de novos veículos de informação, a aplicação de um *web crawler* focado para a busca de notícias, cuja as fontes de dados se apresentam de forma dispersa entre os diferentes veículos de comunicação online, requer uma forma dinâmica para a aquisição das bases de pesquisa, em virtude de que, segundo Furtado et al. (2015), os links encontrados a partir das sementes que não são sub endereços do próprio site, no geral são irrelevantes, como propagandas, entre outros. Se tratando de notícias, um determinado assunto pode estar disponível em diferentes grupos de sites, tornando a prévia informação das sementes algo ineficaz quando se busca um sistema flexível. Aliado a dificuldade de obter as sementes de maneira automática, têm-se o desafio do agente rastreador ser capaz de capturar dentro do corpo da página em análise, somente o que é pertinente a notícia, ou seja, capturar os dados de forma estruturada, como por exemplo reconhecer o que é o título e o conteúdo da matéria, uma vez que cada site possui uma forma diferente de categorizar o conteúdo dos elementos (tags, IDs, classes, entre outros) (SILVA; SANTOS; FERNEDA, 2013).

A partir da identificação de padrões e de requisições a motores de busca, foi possível com a aplicação desenvolvida obter automaticamente as fontes de dados e extrair das páginas analisadas na maioria dos casos somente os dados referentes as notícias.

2. Web crawler

Um *web crawler* - também conhecido como *web spider*, *web bot*, *web indexer*, *web agent* ou *web robot* - é um agente rastreador autônomo que coleta informações de páginas da web, também conhecidas como documentos, baixando seu conteúdo para que este possa ser posteriormente utilizado para os mais diversos propósitos (JASANI; KUMBHARANA, 2014, tradução nossa). *Web crawlers* são flexíveis e podem ter seu comportamento precisamente determinado, sendo um importante método para a coleta de dados e o acompanhamento da rápida expansão da *web* (AHUJA; BAL; VARNICA, 2014, tradução nossa; KUMAR; VIG, 2012, tradução nossa).

2.1 Parser

Segundo Bastos (2006), um dos componente essenciais de um rastreador é o *parser*, um elemento responsável pela classificação do conteúdo baixado de uma página da *web*. Mais especificamente, o *parser* reconhece a estrutura lógica de elementos de uma página, tais como cabeçalho, corpo, *tags*, entre outros, através da análise sequencial de *tokens* do texto do documento em análise e conseqüentemente identifica falhas em sua estrutura (CROFT; METZLER; STROHMAN, 2011, tradução nossa).

Existem diferentes formas e ferramentas para se realizar a análise de documentos, sendo a linguagem de programação Python de grande destaque nesta atividade. Dentre os *parsers* disponíveis na linguagem estão o *html.parser*, *lxml*, e o *html5lib* (RICHARDSON, 2015, tradução nossa). Outro meio para realizar esta tarefa são os *selectors*, um mecanismo presente no framework Scrapy, utilizado para a implementação de *web crawlers*. Os *selectors* possuem funções para a realizar a seleção de partes de um documento HTML ou XML através de expressões Folhas de Estilo em Cascata, do inglês Cascading Style Sheets (CSS) ou XPath, podendo também utilizar expressões regulares de forma conjunta. Sua construção foi baseada no *lxml*, o que resulta em um bom desempenho (SCRAPY..., 2017).

3 Web crawler para extração estruturada de dados

Este projeto se constituiu no desenvolvimento de uma aplicação web para agregar notícias de determinados tópicos informados pelos usuários, e monitorá-los diariamente em busca de novas informações. Para tal, aplicando o conceito de rastreadores focados, foram desenvolvidos dois *web crawlers* que trabalham em conjunto, sendo um para a recuperação das fontes de dados, e outro para a extração dos mesmos. Uma análise nessas fontes foi necessária a fim identificar padrões e obter somente o conteúdo desejado.

Para a execução do projeto, primeiramente se realizou o planejamento da arquitetura, sendo definidas após, as tecnologias necessárias para atendê-la. A linguagem de programação escolhida para o desenvolvimento da parte principal do protótipo foi o Python, tanto para o framework Scrapy quanto para a API REST com o Django REST Framework, que utilizam a linguagem.

3.1 Estudo das ferramentas

No desenvolvimento do protótipo, para a implementação dos *web crawlers*, foi utilizado o *framework Scrapy*, versão 1.4.0. A outra parte do *back-end* da aplicação, responsável por gerenciar as requisições e delegar as chamadas aos rastreadores foi desenvolvida utilizando o *Django REST Framework*, uma biblioteca presente no *Django* que flexibiliza a criação de *Web APIs*. Para a persistência dos dados foi utilizado o *MySQL* e para camada de aplicação se utilizou o *AngularJS*. A versão do Python utilizada foi a 2.7.

Para realizar as chamadas aos *web crawlers* através do *Django REST Framework*, foi utilizado o Scrapyd, uma pequena API que fornece um servidor para que os *spiders* desenvolvidos com o Scrapy possam ser executados via requisições *web*, podendo ser

chamados em paralelo. O Scrapyd oferece também uma interface *web* simples para monitorar os rastreadores e acessar os *logs* de execução, sendo uma ferramenta muito útil na etapa de desenvolvimento.

3.2 Arquitetura

Conforme mencionado, para atender aos objetivos do protótipo, foram implementados dois *web crawlers*, gerenciados em conjunto. O primeiro rastreador tem como propósito obter endereços de notícias, realizando duas tarefas distintas. Na primeira, quando um novo tópico é adicionado pelo usuário, o *crawler* realiza uma requisição ao motor de pesquisa do *Google*, e na sequência faz o *parser* da resposta, capturando somente os *links* das notícias, sendo portanto as sementes.

A segunda tarefa delegada a este rastreador, compreende a etapa diária de revisitação. Nesta, é obtido dos endereços adquiridos anteriormente apenas o domínio, logo a página inicial do site em questão, e em seguida é realizada uma requisição ao mesmo, a fim de obter novas notícias relacionadas ao tópico. Os sites utilizados nas revisitas serão, portanto, sempre os que foram obtidos na primeira tarefa que é a pesquisa do *Google* descrita acima.

Após realizada a primeira etapa, que é a de obter os endereços do tópico informado, ou após a aquisição no processo de revisitação, é disparado uma chamada ao *web crawler* responsável por extrair o conteúdo desses endereços. Na figura 1 é mostrada a arquitetura do protótipo, com os fluxos das comunicações realizadas.

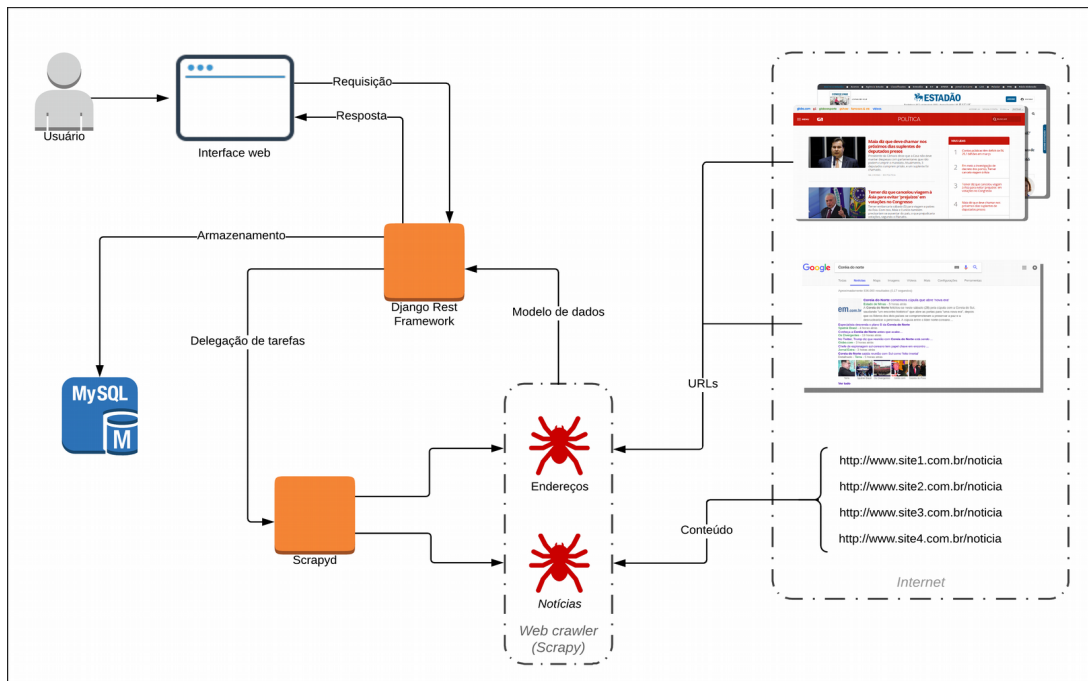


Figura 1. Arquitetura do protótipo

3.3 Web crawler de endereços

Na etapa de obtenção das sementes, é realizado uma requisição ao endereço de pesquisa do *Google* passando dois parâmetros na URL, sendo uma requisição *GET*. O

primeiro parâmetro chamado *q* consiste no termo busca e o outro chamado *tbm* é referente ao filtro. Como o *Google Search* possui uma página dedicada somente a notícias, passando para o *tbm* o valor *nws*, os resultados obtidos são os daquela página.

Tomando como exemplo o termo de busca *incêndio prédio*, a URL resultante seria <https://www.google.com.br/search?q=incêndio+prédio&tbm=nws>. Visando obter uma maior quantidade de resultados, a aplicação foi construída para recuperar todos os endereços encontrados nesta primeira página, uma vez que, como será visto adiante, na extração das notícias são realizadas determinadas validações nos endereços e alguns são descartados.

Com relação à etapa de revisitação, como citado anteriormente, é retirado dos endereços sementes apenas o domínio. Tendo como exemplo a seguinte URL <https://g1.globo.com/economia/noticia/como-dolar-mais-carro-fez-juros-na-argentina-subirem-a-40-e-qual-o-risco-para-o-brasil.ghtml> após a transformação será <https://g1.globo.com/>. Para isso, foi utilizado a biblioteca *urlparse* do Python, que divide uma URL informada em componentes permitindo gerenciá-los separadamente. (URLPARSE, 2018, tradução nossa).

Com os domínios obtidos, deve-se agora capturar todos os *links* referentes ao assunto. Para isso, primeiramente são extraídos todos os *hyperlinks* (*tags <a>*) da página, após isso são realizados dois filtros. O primeiro seleciona somente os endereços de mesmo domínio do site em questão e que possuem o caminho de um recurso especificado, eliminando possíveis referências ao próprio site. Após isso é feita uma comparação do texto contido junto desses *hyperlinks* com o tópico de busca informado pelo usuário, e caso aquele contenha ao menos algumas das palavras deste, então é considerado um endereço pertinente à notícia.

3.4 Extração estruturada de notícias

Para o web crawler de notícias foi necessário montar uma estrutura genérica de extração, uma vez que as fontes de dados não são previamente conhecidas. Na identificação dos padrões encontrados nos sites, com foco para os portais de notícias, notou-se primeiramente a utilização das marcações Open Graph, que são meta tags que otimizam o compartilhamento de conteúdo nas redes sociais, através da identificação das suas partes. Com o Open Graph, pode-se identificar de forma separada dados como o título, descrição, data de publicação de um artigo, além de informações do próprio site (UM GUIA..., 2018).

Desta forma, para a extração do título, descrição, data de publicação e nome do site, utilizou-se os dados disponíveis nessas *meta tags*, uma vez que são disponibilizados em praticamente todos os portais de notícias. Dessas informações, somente o título possui uma alternativa quando não são encontradas as *tags Open Graph*, que seria a própria *tag <title>* do HTML, mas apesar disso, na maioria das vezes, ela traz consigo o nome do site junto do título da matéria.

Para a recuperação do conteúdo da notícia, a implementação baseou-se nas novas *tags* semânticas do HTML5. Com o estudo, chegou-se a uma estrutura geral que os portais de notícias deveriam seguir, e isso foi confirmado após, com a análise comparativa entre eles. Foi realizado portanto a leitura dos dados contidos na *tag <p>* que são filhas da marcação *<article>*, ignorando quaisquer informações pertencentes às *tags <aside>*, *<header>*,

<footer> e <section> que estejam dentro do <article>, como demonstrado no exemplo da figura 2, em que as partes destacadas são as que serão capturadas. Os texto contidos na imagem provém de um gerador automático.

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>...</title>
</head>
<body>
  <p>Mauris eu diam commodo, dignissim nisl quis, egestas tellus. Vivamus sit amet nibh euismod, mollis ante a, commodo risus.</p>
  <article>
    <header>Cras ac scelerisque nibh. Curabitur non fermentum lorem, vel volutpat velit. Sed consectetur leo eget tempor mattis. Sed porttitor varius ullamcorper.</header>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ornare ante ut mollis condimentum. Morbi consectetur massa ac nisi efficitur, et vehicula nunc ultrices. Maecenas vitae ex sem.</p>
    <p>Mauris vel ipsum fermentum, sodales purus ultricies, molestie est. Phasellus pulvinar, lacus in malesuada facilisis, diam enim fringilla magna, non' pulvinar magna libero vel lectus.</p>
    <div>
      <p>Cras eu ex congue, mattis elit nec, condimentum erat. Proin accumsan ullamcorper metus, nec fringilla odio blandit ac. Sed eleifend dictum lorem, eget tincidunt leo suscipit quis.</p>
    </div>
    <aside>
      <p>Nulla nec tempus velit, ut fringilla nisl. Ut orci odio, malesuada eu ullamcorper ac, ullamcorper eget justo. Nam convallis euismod orci, sit amet luctus augue ultricies eu.</p>
    </aside>
    <p>Donec ac arcu scelerisque, molestie arcu vel, interdum nisi. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In tempor, risus vel aliquam vestibulum, magna tellus mattis neque, vel ultrices enim dui nec eros. Vivamus consequat vel quam non blandit. Ut ut ex ante. Ut iaculis ut dolor ut sodales. Aliquam a velit non odio ornare ultrices ac ut tortor.</p>
  </article>
</body>
</html>

```

Figura 2. Exemplo de seções extraídas

A figura 3 apresenta a função que realiza este processo, capturando todos os <article> dentro da tag <body>, sendo o apenas o primeiro utilizado no seletor seguinte. Este por sua vez é encarregado de extrair o texto de todas as marcações <p> seguindo a regra de negação descrita anteriormente. Para esta expressão de negação foi necessária a utilização de um seletor XPath, devido ao fato dos seletores CSS não permitirem a negação de múltiplos elementos de uma só vez.

```

def get_article_p_text(dom):
    article = dom.css('body article')
    p_news = article[0].xpath('.//p[not(ancestor::aside) '
                              'and not(ancestor::header) '
                              'and not(ancestor::footer) '
                              'and not(ancestor::section)]')

    p_contents = []

    for val in p_news:
        p_contents.append(''.join(val.css('*::text').extract()))

    return p_contents

```

Figura 3. Função para extração do conteúdo das notícias

No *pipeline*, após a extração da notícia, é feito uma verificação se foi encontrado dados com a marcação <p> dentro da tag <article> e caso isso não ocorra, os dados da notícia que já foram capturados e seu respectivo endereço são descartados, não sendo a URL utilizada posteriormente no processo de revisitação. Desta forma o critério utilizado para avaliar se o

site oferece ou não uma estrutura semântica na representação da informação é baseado nas informações contidas na marcação *<article>*

Na figura 4 é apresentado a parte de interação com o usuário do protótipo, sendo a listagem da esquerda referente ao tópicos escolhidos até o momento e na direita a relação de notícias do tópico selecionado. Nesta última são exibidos os dados obtidos nas meta *tags Open Graph*.



Figura 4. Listagem de tópicos e notícias do protótipo

Ao clicar em um item da listagem é aberto o detalhamento da notícia, como apresentado na figura 5. Nesta tela são exibidos o título, o conteúdo que foi extraído da *tag <article>* e no rodapé a fonte (nome do site) com endereço para o *link* original da matéria em questão.

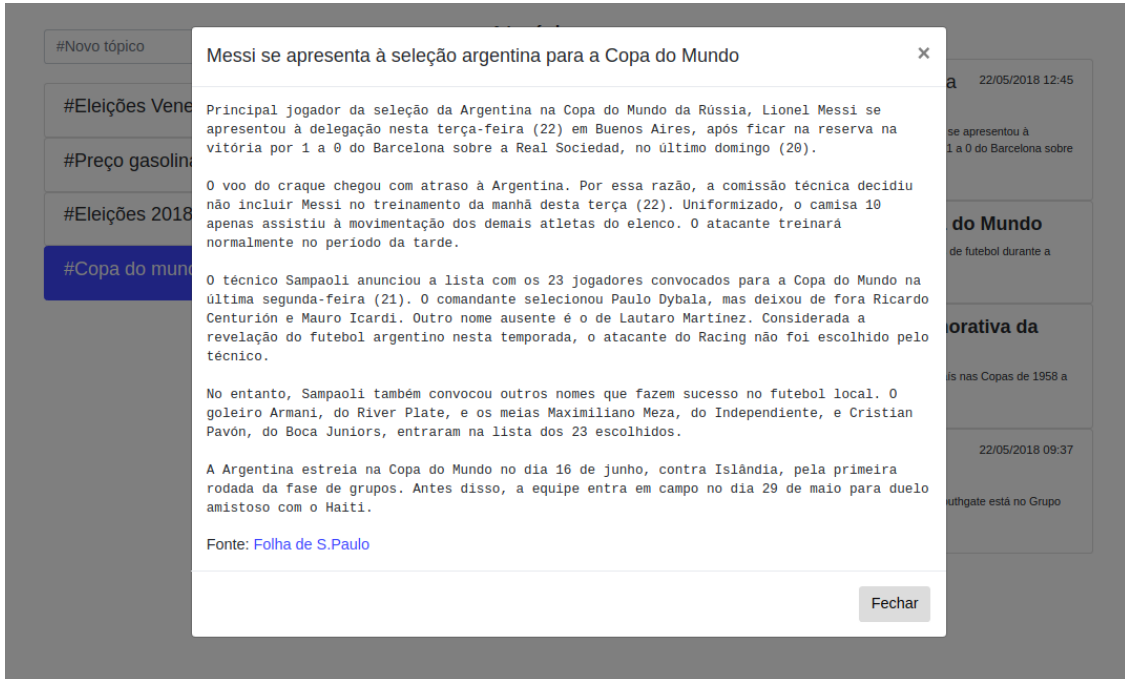


Figura 5. Detalhamento de uma notícia

3.5 Resultados

Tratando-se da obtenção das fontes de dados, ou seja, das sementes do *web crawler*, os resultados foram os esperados, sendo todos relacionados com o tópico informado, dado a eficiência de indexação das páginas pelo motor de busca do Google. Com relação a extração da notícia, o protótipo realizou somente a extração do texto puro, sem a inserção de links e ignorando imagens, vídeos e outras estruturas como tabelas. Sendo assim, os resultados obtidos com a implementação genérica apresentada na seção 3.4, no qual não são levados em consideração sites sem dados nas *tags* `<p>` filhas da marcação `<article>`, foram em sua maioria satisfatórios, trazendo nesses casos somente o conteúdo relacionado a notícia conforme já apresentado na figura 5. Apesar disso, em algumas vezes, o conteúdo obtido continha outras informações que não tinham relação direta com o artigo.

Isso ocorreu por dois motivos. O primeiro é que realmente existem em alguns sites, textos disponíveis dentro da *tag* `<article>` sem qualquer indicação de que não fazem parte do conteúdo principal, diferente do que é encontrado na maioria dos portais, no qual informações semelhantes estavam dentro da marcação `<aside>`, o que seria o ideal nesses casos. O segundo motivo, de menor ocorrência, é relacionado ao próprio uso de um *web crawler* para acessar estes sites, pois muito dos textos extraídos tinham relação com problemas de *scripts* da página, entre outros, mensagens estas que não apareceriam caso o site estivesse sendo acessado por navegador tradicional. Outra limitação tem relação com as legendas de fotos e vídeos, pois com a estratégia de extração desenvolvida a mídia correspondente não é incorporada, ficando um texto aparentemente sem sentido, porém que faz parte da notícia. Subtítulos que não estavam dentro das *tags* de parágrafo também não foram extraídos.

Com isso, foi verificado que dos sites que possuíam a *tag* `<article>` com marcações de parágrafo, todos continham dentro dela o conteúdo completo da notícia, só havendo a falta na indicação de conteúdos não relacionados.

4. Conclusão

Com a *web* em crescente expansão e com sua dinamicidade na geração e atualização de novas informações, a utilização de ferramentas que facilitam o consumo dessa massa de dados se torna indispensável. Neste contexto, entra em cena o uso de agentes autônomos para capturar e gerir esses dados. A utilização desses agentes se mostra de grande valor não somente para o auxílio no consumo de informação, mas também para outras áreas importantes como o *Data Mining*.

A obtenção desses dados de maneira automática e estruturada implica porém, que essas informações sejam legíveis não somente para humanos, mas também para as máquinas, ou seja, distribuídas de forma semântica. Com a expansão da *web* essa preocupação foi crescendo, sendo apresentado já em 2001 o conceito da Web Semântica e seus objetivos para uma melhor distribuição dos dados na internet. Com este ideal, as tecnologias utilizadas para o desenvolvimento *web* estão evoluindo cada vez mais para que exista uma forma comum na definição das informações. Um exemplo disto é o HTML5, no qual introduziu marcações semânticas que possibilitaram no protótipo deste trabalho a extração estruturada do conteúdo da notícia. O advento das redes sociais fez também surgir soluções para um melhor compartilhamento de conteúdo nesses meios, como as meta *tags Open Graph*, utilizadas, ainda que as vezes de forma incompleta, por todos os portais de notícias que o protótipo analisou, e que se mostraram eficientes para a extração de metadados pelo *web crawler* desenvolvido.

Portanto, com esse estudo, foi possível identificar que as tecnologias envolvidas no desenvolvimento *web* estão de fato tendo a preocupação da interoperabilidade, disponibilizando e aplicando técnicas para dar significado aos dados, embora o conceito de Dados Conectados, como por exemplo a utilização de RDF para descrição de recursos e de suas relações, não seja totalmente aplicado ainda.

Apesar disso, mesmo com o que as tecnologias oferecem no momento, foi visto que muitos sites ainda não utilizam todos os recursos disponíveis ou não utilizam da forma correta. Existem ainda muitos portais que não fazem o uso correto das novas *tags* HTML5 para delimitar o conteúdo principal, fazendo com o que uma possível extração dinâmica nesses casos seja muito custosa, uma vez que quando não existe uma marcação explícita, a única forma de obter o conteúdo desejado é somente através de técnicas de comparação avançada, o que por vezes resultaria em uma baixa precisão nos resultados, dado a gama de possibilidades de informações e estruturas existentes ao se trabalhar com fontes desconhecidas. Mesmo que se alcance bons resultados com este método, ainda assim a informação não estaria em conformidade com o conceito da Web Semântica, que visa avanços na forma de distribuir os dados.

A respeito da abordagem de implementação dos *web crawlers*, dentre as políticas aplicadas neste protótipo, a de maior destaque é a política de seleção, pois com os métodos utilizados não foi necessário em nenhum dos casos a busca em profundidade, como pode-se ver primeiramente no processo de obtenção das bases de pesquisa e posteriormente na etapa de revisitação. No primeiro, isso ocorreu devido aos resultados do motor de busca do Google serem precisos e já direcionados para a página da notícia. Aqui foi utilizado de forma indireta

a métrica do *PageRank*, uma vez que este sistema para classificação das páginas está incorporado no mecanismo. Com relação ao segundo processo, a etapa de revisitação, naturalmente as últimas notícias tendem a ficar na página inicial, sendo mais uma vez desnecessário a busca em profundidade.

Referências

AHMADI-ABKENARI, F.; SELAMAT, Ali. A Clickstream-based Focused Trend Parallel Web Crawler. **International Journal Of Computer Applications**. Bangalore, p. 1-8. nov. 2010. Disponível em: <<http://www.ijcaonline.org/volume9/number5/pxc3871866.pdf>>. Acesso em: 05 maio 2017.

AHUJA, Mini Singh; BAL, Jatinder Singh; VARNICA. Web Crawler: Extracting the Web Data. **International Journal of Computer Trends and Technology**. Thennur, p. 132-137. jul. 2014. Disponível em: <<http://www.ijcttjournal.org/Volume13/number-3/IJCTT-V13P128.pdf>>. Acesso em: 11 junho 2017.

BASTOS, Valeria Menezes. **Ambiente de descoberta de conhecimento na web para a língua portuguesa**. 2006. 124 f. Tese (Doutorado) - Curso de Ciências em Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006. Disponível em: <http://www.poc.ufrj.br/teses/doutorado/inter/2006/Teses/BASTOS_VM_06_t_D_int.pdf>. Acesso em: 07 maio 2017.

CROFT, W. Bruce; METZLER, Donald; STROHMAN, Trevor. **Search Engines: Information Retrieval in Practice**. [s.i]: Pearson Education, 2011. 552 p.

JASANI, Bhavin M.; KUMBHARANA, C. K.. Analyzing Different Web Crawling Methods. **International Journal of Computer Applications**, Bangalore, v. 107, n. 5, p.23-26, 05 dez. 2014. Disponível em: <<http://research.ijcaonline.org/volume107/number5/pxc3900000.pdf>>. Acesso em: 14 ago. 2017.

KUMAR, Mukesh; VIG, Renu. Learnable Focused Meta Crawling Through Web. In: INTERNATIONAL CONFERENCE ON COMMUNICATION, COMPUTING & SECURITY, 2., 2012, Rourkela. **Proceedings...** Chandigarh: Procedia Technology, 2012. p. 606 - 611. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2212017312006184>>. Acesso em: 14 maio 2017.

RICHARDSON, Leonard. **Beautiful Soup Documentation**. [2015]. Disponível em: <<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>>. Acesso em: 02 set. 2017.

SCRAPY Documentation: Release 1.5.0. Release 1.5.0. 2017. Disponível em: <<https://media.readthedocs.org/pdf/scrapy/latest/scrapy.pdf>>. Acesso em: 20 jan. 2018.

SILVA, Renata Eleuterio da; SANTOS, Plácida Leopoldina V. A. da Costa; FERNEDA, Edberto. Modelos de recuperação de informação e web semântica: A questão da relevância. **Informação & Informação**, [s.l.], v. 18, n. 3, p.27-44, 9 out. 2013. Universidade Estadual de Londrina. <http://dx.doi.org/10.5433/1981-8920.2013v18n3p27>. Disponível em: <http://www.uel.br/revistas/uel/index.php/informacao/article/viewFile/12822/pdf_3>. Acesso em: 10 nov. 2017.

UMGUIA de compartilhamento para Webmasters. 2018. Disponível em: <https://developers.facebook.com/docs/sharing/webmasters?locale=pt_BR#markup>. Acesso em: 12 maio 2018.

URLPARSE: Parse URLs into components. Parse URLs into components. 2018. Disponível em: <<https://docs.python.org/2/library/urlparse.html#module-urllparse>>. Acesso em: 05 maio 2018.