Article
Yuri Marx · Dec 23, 2020 6m read

Open Exchange

Technical View: Crawler and NLP to do website text analytics

Web Crawling is a technique used to extract root and related content (HTML, Videos, Images, etc.) from websites to your local disk. This is allows you apply NLP to analyze the content and get important insights. This article detail how to do web crawling and NLP.

To do web crawling you can choose a tool in Java or Python. In my case I'm using Crawler4J. (https://github.com/yasserg/crawler4j).

Crawler4j is an open source web crawler for Java which provides a simple interface for crawling the Web. Using it, you can setup a multi-threaded web crawler in few minutes.

After create your Java project, include maven dependency:

```
<dependency>
        <groupId>edu.uci.ics</groupId>
        <artifactId>crawler4j</artifactId>
        <version>4.4.0</version>
    </dependency>
```

Now, create a class to do the crawler. See you need extends WebCrawler and implement shouldVisit to indicate the file types that will be extracted and visit to get the content and write in your disk. Very simple!

```
public class CrawlerUtil extends WebCrawler {

    private final static Pattern FILTERS = Pattern.compile(".*(\\.(css|js|gif|jpg|png
|mp3|mp4|zip|gz))$");

    @Override
    public boolean shouldVisit(Page referringPage, WebURL url) {
        System.out.println("shouldVisit: " + url.getURL().toLowerCase());

        String href = url.getURL().toLowerCase();
        boolean result = !FILTERS.matcher(href).matches();

        if (result)
            System.out.println("URL Should Visit");
        else
            System.out.println("URL Should not Visit");

        return result;
    }

    @Override
    public void visit(Page page) {
        String url = page.getWebURL().getURL();
        System.out.println("URL: " + url);

        if (page.getParseData() instanceof HtmlParseData) {
```

```
            HtmlParseData htmlParseData = (HtmlParseData) page.getParseData();
            String text = htmlParseData.getText(); //extract text from page
            String html = htmlParseData.getHtml(); //extract html from page
            Set<WebURL> links = htmlParseData.getOutgoingUrls();

            System.out.println("------------------------------------------------------
----");
            System.out.println("Page URL: " + url);
            System.out.println("Text length: " + text.length());
            System.out.println("Html length: " + html.length());
            System.out.println("Number of outgoing links: " + links.size());
            System.out.println("------------------------------------------------------
----");

            final String OS = System.getProperty("os.name").toLowerCase();

            FileOutputStream outputStream;

            File file;

            try {

                if(OS.indexOf("win") >= 0) {
                    file = new File("c:\\crawler\\nlp" + UUID.randomUUID().toString()
 + ".txt");
                } else {
                    file = new File("/var/crawler/nlp/" + UUID.randomUUID().toString(
) + ".txt");
                }

                outputStream = new FileOutputStream(file);

                byte[] strToBytes = text.getBytes();

                outputStream.write(strToBytes);

                outputStream.close();

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

After, you need to create a Java PEX Business Operation to allows IRIS call this Crawler implementation into a production. To do this is necessary extends the class BusinessOperation (from intersystems) and write your code, simple! The method OnInit allows you instantiate the Java Gateway (a middleware from the InterSystems to promote ObjectScript and Java conversation). In the method OnMessage, you use gateway instance into iris variable to send the java results to IRIS.

The Crawler Java logic in this class run in the executeCrawler. You need set setMaxDepthOfCrawling to config how many sub pages will be visited and setCrawlStorageFolder with folder path to store Crawler4J processing. So you call start passing the number of pages, to limit your crawling. The robotconfig allows you follow the website policies about crawling and another rules. See all the code:

```
public class WebsiteAnalyzerOperation extends BusinessOperation {
```

```
    // Connection to InterSystems IRIS
    private IRIS iris;

    @Override
    public void OnInit() throws Exception {
        iris = GatewayContext.getIRIS();
    }

    @Override
    public Object OnMessage(Object request) throws Exception {
        IRISObject req = (IRISObject) request;

        String website = req.getString("Website");
        Long depth = req.getLong("Depth");
        Long pages = req.getLong("Pages");

        String result = executeCrawler(website, depth, pages);
        IRISObject response = (IRISObject)(iris.classMethodObject("Ens.StringContaine
r","%New", result));

        return response;
    }

    public String executeCrawler(String website, Long depth, Long pages) {

        final int MAX_CRAWL_DEPTH = depth.intValue();
        final int NUMBER_OF_CRAWELRS = pages.intValue();

        final String OS = System.getProperty("os.name").toLowerCase();

        String CRAWL_STORAGE = "";

        if(OS.indexOf("win") >= 0) {
            CRAWL_STORAGE = "c:\\crawler\\storage";
        } else {
            CRAWL_STORAGE = "/var/crawler/storage";
        }

        CrawlConfig config = new CrawlConfig();
        config.setCrawlStorageFolder(CRAWL_STORAGE);
        config.setIncludeHttpsPages(true);
        config.setMaxDepthOfCrawling(MAX_CRAWL_DEPTH);

        PageFetcher pageFetcher = new PageFetcher(config);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFe
tcher);

        CrawlController controller;
        try {
            controller = new CrawlController(config, pageFetcher, robotstxtServer);

            controller.addSeed(website);

            controller.start(CrawlerUtil.class, NUMBER_OF_CRAWELRS);

            return "website extracted with success";

        } catch (Exception e) {
```

```
            return e.getMessage();
        }

    }

    @Override
    public void OnTearDown() throws Exception {

    }

}
```

With the content extracted, you can create a NLP domain pointing to the folder with the extractions, see:

```
<files listname="File_1" batchMode="false" disabled="false" listerClass="%iKnow.Sourc
e.File.Lister" path="/var/crawler/nlp/" recursive="false" extensions="txt">
<parameter value="/var/crawler/nlp/" isList="false" />
```

All the implementation:

```
Class dc.WebsiteAnalyzer.WebsiteAnalyzerNLP Extends %iKnow.DomainDefinition [ Procedu
reBlock ]
{

XData Domain [ XMLNamespace = "http://www.intersystems.com/iknow" ]
{
<domain name="WebsiteAnalyzerDomain" disabled="false" allowCustomUpdates="true">
<parameter name="DefaultConfig" value="WebsiteAnalyzerDomain.Configuration" isList="f
alse" />
<data dropBeforeBuild="true">
<files listname="File_1" batchMode="false" disabled="false" listerClass="%iKnow.Sourc
e.File.Lister" path="/var/crawler/nlp/" recursive="false" extensions="txt">
<parameter value="/var/crawler/nlp/" isList="false" />
<parameter isList="false" />
<parameter value="0" isList="false" />
</files>
</data>
<matching disabled="false" dropBeforeBuild="true" autoExecute="true" ignoreDictionary
Errors="true" />
<metadata />
<configuration name="WebsiteAnalyzerDomain.Configuration" detectLanguage="true" langu
ages="en,pt,nl,fr,de,es,ru,uk" userDictionary="WebsiteAnalyzerDomain.Dictionary#1" su
mmarize="true" maxConceptLength="0" />
<userDictionary name="WebsiteAnalyzerDomain.Dictionary#1">
</domain>
}

}
```

Now, InterSystems NLP can present the results to you into Text Analytics - Domain Explorer.

See all implementation code in the github, download and run!

#Analytics #Contest #Java #InterSystems IRIS
Check the related application on InterSystems Open Exchange